LANGAGE ET DÉVELOPPEMENT

RSX ET ROUTINES ASSEMBLEUR SUR AMSTRAD CPC

DIDIER ROY
JEAN-JACQUES WEYER

RSX ET ROUTINES ASSEMBLEUR SUR AMSTRAD CPC

CONNAISSEZ-VOUS TOUTE LA COLLECTION AMSTRAD CHEZ P.S.I. ?

Pour les Amstrad CPC 464, 664 et 6128 :

Initiation :

— La découverte de l'Amstrad - Daniel-Jean David
— Exercices en BASIC pour Amstrad - Maurice Charbit

Programmation BASIC :

— 102 programmes pour Amstrad - Jacques Deconchat
— Super jeux pour Amstrad - Jean-François Sehan
— Amstrad en famille - Jean-François Sehan
— Super générateur de caractères sur Amstrad - Jean-François Sehan
— Photographie sur Amstrad et Apple II - Pierrick Moigneau et Xavier de la Tullaye
— Amstrad en musique - Daniel Lemahieu
— Amstrad à l'école - Daniel Nielsen et Augustin Garcia Ampudia

Maîtrise du BASIC :

— BASIC Amstrad - 1. Méthodes pratiques - Jacques Boisgontier et Bruno Césard
— BASIC Amstrad - 2. Programmes et fichiers - Jacques Boisgontier
— BASIC Plus : 80 routines sur Amstrad - Michel Martin
— Périphériques et fichiers sur Amstrad - Daniel-Jean David

Assembleur et Pascal :

— Assembleur de l'Amstrad - Marcel Henrot
— Graphisme en assembleur sur Amstrad CPC - Francis Piérot
— Création et animations graphiques sur Amstrad CPC - Gilles Fouchard et Jean-Yves Corre
— Trois étapes vers l'intelligence artificielle sur Amstrad CPC - René Descamps
— IA sur Amstrad CPC, langages et formes - Eric et Thierry Lévy-Abegnolli
— Turbo Pascal sur Amstrad - Pierre Brandeis et Frédéric Blanc
— Amstrad 3-D - Jean-Pierre Peht

Système :

— Clefs pour Amstrad - 1. Système de base - Daniel Martin
— CP/M Plus sur Amstrad 6128 et 8256 - Yvon Dargery
— Clefs pour Amstrad - 2. Système disque - Daniel Martin et Philippe Jadoul
— Gestion sur Amstrad PCW - Jean-Michel Jego et Alain Gargadennec

A paraître :

— Clefs pour Amstrad PCW - Didier Roy et Jean-Jacques Weyer

Pour tout problème rencontré dans les ouvrages P.S.I. vous pouvez nous contacter au numéro ci-dessous :

Numéro Vert/Appel Gratuit en France

05 21 22 01

(Composer tous les chiffres, même en région parisienne)

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective », et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite » (alinéa 1er de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

© *Éditions du P.S.I. - B.P. 86 - 77402 Lagny/Marne cedex*
1986

ISBN 2-86595-352-1

LANGAGE ET DÉVELOPPEMENT

RSX ET ROUTINES ASSEMBLEUR SUR AMSTRAD CPC

DIDIER ROY
JEAN-JACQUES WEYER

ÉDITIONS DU P.S.I.
1986

PRÉSENTATION DES AUTEURS

A Marie-Noëlle et Odile
pour leur infinie patience.

Didier Roy

Informaticien au Centre national informatique de l'Institut national de la statistique et des études économiques. Après des études supérieures d'informatique et d'électronique, il se passionne pour le monde de la micro-informatique et le jeu d'échecs depuis une dizaine d'années.

Il est auteur d'un logiciel de prévision statistique du Loto sur ORIC et AMSTRAD.

Jean-Jacques Weyer

Professeur de mathématiques, et responsable de la section EAO (Enseignement Assisté par Ordinateur) de son établissement.

Depuis deux ans, il se passionne pour les micro-ordinateurs, ce qui lui a permis de faire la connaissance de Didier Roy.

Ces deux dernières années, ils ont ensemble collaboré à un premier ouvrage sur l'Amstrad « LES ROUTINES DE L'AMSTRAD CPC 464, 664, ET 6128 », qui paraîtra en novembre 85 chez MICRO-APPLICATION. Ils préparent actuellement un troisième ouvrage sur les logiciels des PCW 8256 et 8512.

Amstrad CPC 464-664 et 6128 sont des marques déposées de Amstrad Consumer Electronics.

DEVPAC est une marque déposée de HiSoft et Amstrad Consumer Electronics plc.

ZEN est une marque déposée de Amstrad Consumer Electronics.

DAMS est une marque déposée de Micro-Application.

SOMMAIRE

INTRODUCTION À L'OUVRAGE 9

Objet 9
Conception de l'ouvrage 9
Présentation des chapitres 9
Mise en pratique des routines 10

CHAPITRE 1 : RUDIMENTS D'ASSEMBLEUR Z80 13

Préliminaire 13
Structure d'un programme assembleur 15
Les pseudo-opérations 17

Les instructions assembleur 19
Les registres du Z80 20
Fonctions des bits du registre d'état 21
Test des valeurs prises par les indicateurs 22
Les instructions du Z80 22

Modification des indicateurs du registre d'état 41
Légendes utilisées 41

CHAPITRE 2 : TRAITEMENT DES NOMBRES 45

Préliminaire 45
Premiers systèmes de numérotation simple à traiter en assembleur 46
Vecteurs mathématiques des CPC 464, 664 et 6128 47

Routines de traitement des nombres entiers 16 bits 50
Routines de traitement des nombres flottants 80
Routine d'affichage d'un flottant en décimal 83
Routine de récupération de la partie entière d'un flottant 93
Routine de récupération de la partie décimale d'un flottant 105

CHAPITRE 3 : TRAITEMENT DES LISTES 117

Préliminaire 117
Recherche d'une chaîne dans une zone donnée 118
Routine de remplacement d'un texte par un autre 123
Routines de tri de tableaux 137
Routine de tri de tableau de chaînes 141
Routine de tri de tableaux d'entiers 151
Routine de tri de tableaux de réels 151

CHAPITRE 4 : GRAPHISMES 165

Généralités 165

Extension Basic pour CPC 464 166
Couleurs graphiques 166
Fenêtres graphiques 170

Tracé de figures géométriques 174
Généralités 174
Tracé de triangle 174
Tracé de rectangle 181
Tracé de rectangle (avec orientation) 187
Tracé de cercle 196
Tracé d'ellipse (avec orientation) 204
Tracé de polygone régulier (avec orientation) 215

Traitement de la mémoire écran 224
Généralités 224
Copie d'une zone de l'écran 227
Zoom d'une partie de l'écran 234
Super-zoom 244

CHAPITRE 5 : MATHÉMATIQUES 255

Généralités 255

Aires 257
Volumes 265
Électricité 274
Physique 283
Mathématiques 293

ANNEXE 1 : UTILISATION, IMPLANTATION, LONGUEUR DES ROUTINES 301

Nom et fonction des routines non RSX 301
Utilisation des routines non RSX 302
Nom et fonction des routines RSX 302
Appel RSX avec passage de paramètres 303
Tableau d'implantation des routines en mémoire 304
Tableau des longueurs 305

ANNEXE 2 : LES CHARGEURS DE ROUTINES 307

Précautions importantes 308
Précision concernant le contrôle 308

ANNEXE 3 : VECTEURS SYSTÈMES 351

Vecteurs mathématiques 351
Vecteurs d'affichage 351
Vecteurs clavier 351
Vecteurs graphiques 352

ANNEXE 4 : LEXIQUE 355

ANNEXE 5 : LA DISQUETTE PROGRAMME 361

CONSEILS DE LECTURE 365

INTRODUCTION A L'OUVRAGE

OBJET

Cet ouvrage présente 30 routines, dont la plupart peuvent être appelées par une instruction RSX.

L'objectif visé dans cet ouvrage est subdivisé en trois pôles :

- Apprentissage progressif de l'assembleur Z80 au travers des routines assembleur présentées.
- Découvertes des dessous de l'informatique : les méthodes de programmation couramment utilisées en informatique, et bien sûr en micro-informatique.
- Technicité et utilité des routines dans divers domaines de la micro-informatique.

CONCEPTION DE L'OUVRAGE

Pour chaque chapitre traitant d'un sujet spécifique, un certain nombre de routines assembleur sont traitées.
Pour chaque routine sont présentés : la fonction, la méthode de traitement, un organigramme de programmation, le programme en assembleur dûment documenté pour une meilleure compréhension, et éventuellement un ou plusieurs courts programmes BASIC effectuant une démonstration sur l'utilisation de la routine.

PRÉSENTATION DES CHAPITRES

Le lecteur peut aborder l'ouvrage par n'importe lequel des chapitres, mais si ses connaissances en assembleur Z80 ne sont pas très étendues, nous lui recommandons de commencer par le premier chapitre qui traite des instructions assembleur Z80.

Le premier chapitre présente successivement, et dans cet ordre, les pseudo-instructions, les instructions, puis un tableau sur la modification des indicateurs après exécution de chacune des instructions assembleur.

Avec le chapitre 2, le lecteur découvrira le traitement interne des nombres en mémoire, permettant ainsi au micro-ordinateur de réaliser les diverses conversions et affichages de nombres en une base 2, 10, ou 16, qu'ils soient signés ou non, entiers ou réels.

Le chapitre 3, quant à lui, est orienté vers la gestion et le traitement de texte, puisqu'il vous permettra d'effectuer la recherche de texte dans une liste, avec ou sans changement de celui-ci, puis de réaliser le tri de n'importe quel type de tableau, qu'il soit défini en Basic ou en assembleur.

Pour ce qui concerne le chapitre 4, il est scindé en deux parties distinctes :

— tracé de formes diverses et utiles, avec pour certaines la possibilité de les orienter dans l'espace écran ;
— réalisation de fonctions manquantes pour le CPC 464, puis présentation de routines de copie et de zoom de zones écran.

Dans le chapitre 5, nous avons voulu fournir une panoplie des possibilités de traitement de fonctions mathématiques, et physiques, dont l'emploi et la réalisation sont aisées.

MISE EN PRATIQUE DES ROUTINES

Les routines assembleur sont présentées par l'intermédiaire d'un assembleur-désassembleur.
Vous disposez donc d'une liste correspondante à une source, et qui est donc facile à saisir sur votre Amstrad, si vous possédez ce type de logiciel.
Si ce n'est pas le cas, il est également très facile, avec un petit chargeur BASIC, de saisir le code machine correspondant. En effet, la première colonne vous indique les adresses ou sont implantés les codes hexa, puis la deuxième colonne vous en donne la liste.
Toutefois, il vous faudra entrer séparément les messages éventuels de la routine, en tenant compte des adresses d'implantation.

Exemple de chargeur de codes

```
INPUT « adresse début : » ; adrdeb : INPUT « adresse fin : » ; adrfin
FOR i=adrdeb TO adrfin : READ code : POKE i, code : NEXT
DATA ..................... code à saisir .....................
```

Exemple de chargeur de message

```
10 INPUT « adresse début : » ; adrdeb
20 INPUT « texte : » ; txt$
30 long=LEN (txt$) : j=1
40 FOR i=adrdeb TO adrdeb+long-1
50 c=ASC(mid$(txt$,j,1))
60 POKE i,c : j=j+1
70 NEXT
80 i=i+1 : POKE i,0
90 GOTO 10
```

La routine d'affichage utilisée dans les routines a besoin d'un zéro binaire en fin de message ; c'est la raison pour laquelle il faut mettre l'instruction 80.

Appel des routines assembleur

Le chapitre sur le traitement des nombres est essentiellement destiné à une utilisation assembleur ; aussi l'appel se fera par un CALL classique ASSEMBLEUR.
En ce qui concerne le passage des valeurs entières ou flottantes, il s'effectuera par l'utilisation d'instruction de chargement ; il y aura donc lieu de s'enquérir de l'adresse d'implantation de la valeur entière ou flottante que la routine connaît.

Pour les autres chapitres, l'appel se fera par une instruction de type RSX, avec passage d'un ou plusieurs paramètres.
RSX vient de l'anglais Resident System eXtensions, soit extension d'instructions BASIC résidentes en mémoire RAM (ou utilisateur). Cette possibilité permet d'augmenter la puissance du BASIC actuel des Amstrad CPC par ajout de nouvelles instructions.

Pour le chapitre sur les routines mathématiques et physiques, il y aura des valeurs à récupérer en sortie ; ceci se fera avec les premiers paramètres qui auront servi pour l'entrée.

Relogeabilité des routines

Les routines sont entièrement relogeables, par simple changement de l'adresse d'implantation (voir pseudo-instruction ORG), dans le cas ou vous utilisez un assembleur-désassembleur.
Si vous ne disposez que d'un chargeur BASIC, l'adresse de chaque donnée étant incluse dans les codes hexa, les données seront figées à l'emplacement prévu actuellement ; mais rien ne vous empêche de les changer au niveau du code hexa...
En ANNEXE 2, vous trouverez la liste des chargeurs des routines présentées dans cet ouvrage.

CHAPITRE 1

RUDIMENTS D'ASSEMBLEUR Z80

PRÉLIMINAIRE

Une programmation en langage machine apporte dans la majorité des cas une puissance d'exécution que nul autre langage ne permet. Tous les autres langages nécessitent une ou plusieurs traductions avant d'être directement exécutables par le microprocesseur.

Le seul langage disponible immédiatement lors du branchement de votre Amstrad CPC est un langage Basic résident.
Ce langage est relativement facile à apprendre — il est assez proche du langage humain — et s'adapte bien aux applications dites mathématiques.
Mais ce qui est facile en informatique coûte souvent fort cher, non pas en espèces sonnantes et trébuchantes, mais en temps ou en place, ou les deux, ce qui est pire...

Prenons un court programme Basic pour illustrer ces propos :

```
10 FOR A=10 TO 0 STEP -1
20 X=X+1
30 NEXT
```

La représentation en mémoire de ce programme est la suivante :

17 00 0A 00 9E 20 0D 00 00 E1 EF 19 0A 20 EC 20 0E 20 E6 20 F5 0F 00 10 00 14 00 0D 00 00 F8 EF 0D 00 00 F8 F4 0F 00 06 00 1E 00 B0 00 00 00 00 00

Cela fait un nombre d'octets égal à 49, et de plus, ces codes ne sont pas directement exécutables par le processeur ; en effet, chaque instruction est interprétée à chaque fois que le programme en demande l'exécution ; ce processus est très coûteux en temps de traitement, et, nous allons le voir, en place mémoire.

```
10 INPUT "adress debut:";adrdeb
20 INPUT "texte:";txt$
30 long=LEN(txt$):j=1
40 FOR i=adrdeb TO adrdeb+long-1
50 c=ASC(MID$(txt$,j,1))
60 POKE i,c:j=j+1
70 NEXT
80 i=i+1:POKE i,0
90 GOTO 10
```

```
10 FOR a=10 TO 0 STEP -1
20 x=x+1
30 NEXT
```

Voyons maintenant la programmation équivalente en langage assembleur Z80 :

```
7000              10          ent # 7000     ; adresse exécution
7000              20          org # 7000     ; début routine
7000   0E00       30          ld c,0         ; init c à 0
7002   3E0B       40          ld a,11        ; init a à 11
7004   0C         50 som :    inc c          ; + 1 dans c
7005   3D         60          dec a          ; − 1 dans a
7006   20FC       70          jr nz, som     ; aller à som si a < > 0
7008   79         80          ld a,c         ; c dans a
7009   320D70     90          ld (×),a       ; a dans ×
700C   C9        100          ret            ; fin routine
700D             110 × :      defs 1         ; init × a 0
```

Ce programme assembleur fait 13 octets contre 49 en Basic, et ce, pour effectuer le même traitement ; de plus, les codes machine sont immédiatement exécutés par le processeur, d'où un gain de temps considérable.

La représentation en mémoire de ce programme se résume donc à ces quelques octets :

0E 00 3E 0B 0C 3D 20 FC 79 32 0D 70 C9

Bien entendu, il existe des logiciels appelés compilateurs, et qui ont pour but de transformer un langage évolué en langage machine. Ce type de logiciel est très gourmand en place mémoire ; de plus, les programmes une fois compilés occupent généralement une place mémoire qui est bien plus importante qu'un programme équivalent écrit directement en langage machine.

En résumé, on peut admettre qu'en général, ni les interpréteurs ni les compilateurs ne permettent d'accéder à une réelle optimisation en temps d'exécution et en place mémoire, du moins sur un micro-ordinateur...
Mais n'allez pas conclure que l'assembleur est la panacée des langages, car s'il offre un résultat satisfaisant en temps et en place, il n'en est pas de même pour l'écriture et la maintenance d'un tel langage.

Mais pas de panique, comme tout langage, il est indispensable de connaître un certain nombre de concepts, que nous allons tenter de découvrir ensemble dans les pages qui vont suivre.

Le court programme assembleur présenté ci-dessus nous amène à deux remarques importantes :

Première remarque

Tout programme assembleur peut être directement chargé par les codes machine ; dans ce cas, il faut disposer d'un chargeur, et il faut surtout posséder une excellente connaissance de ces codes. Si je vous dis de prime abord 79, combien d'entre vous me diront qu'il s'agit de l'instruction assembleur LD A,C qui a pour but de charger le contenu du registre C dans le registre A ?...
Et pourtant, c'est le seul langage que connaît l'ordinateur !
Toutefois, si vous disposez d'un chargeur de codes machine, ce sont bien ces codes qu'il faudra indiquer dans vos lignes de DATA ou dans vos instructions POKE.
Le programme ainsi obtenu constitue le « code objet ».

Deuxième remarque

Si vous utilisez un assembleur-désassembleur, il vous suffira alors d'entrer des instructions telles que LD A,C ; après avoir tapé toutes vos instructions assembleur, il vous sera facile de les faire transcrire en codes machine.
Le programme ainsi obtenu constitue le « code source » ; c'est à partir de celui-ci que toutes modifications éventuelles devront être réalisées.
Pour tous ceux qui veulent véritablement faire de l'assembleur, je ne peux que leur conseiller de se procurer un logiciel qui offre une réelle programmation en assembleur.

Il existe à l'heure actuelle de bons logiciels répondant aux besoins du programmeur en assembleur Z80.

Nous pouvons en citer quelques-uns :

— DEVPAC ASSEMBLER → de chez HISOFT
— ZEN → de chez AMSOFT
— DAMS → de chez MICRO-APPLICATION

Les programmes assembleur présentés dans cet ouvrage ont été développés à l'aide du DEVPAC ASSEMBLER de chez HISOFT.

STRUCTURE D'UN PROGRAMME ASSEMBLEUR

Nous allons maintenant tenter de voir ce qu'est la programmation en assembleur Z80, et en quoi elle diffère de la programmation en Basic.
Prenons un exemple simple : l'affichage d'un texte ; en Basic, cela se fera simplement par l'ordre suivant :

```
10 PRINT « TEXTE A AFFICHER »
```

Comme tout utilisateur, nous sommes satisfaits de nous, et du reste, cela est tout à fait légitime, puisqu'avec une simple instruction nous obtenons le résultat escompté.

En revanche, ce qu'il faut savoir, dans une optique d'approche du langage assembleur, c'est que cette simple instruction est soumise à un logiciel qui s'appelle « INTERPRETEUR BASIC ».
Sa fonction est d'interpréter chaque instruction Basic au moment ou ladite instruction doit être exécutée.

Mais que fait donc cet interpréteur pour arriver au résultat tant attendu ?

Pour chacun des caractères du texte, il va fournir une adresse à une routine qui procédera à l'affichage de ce caractère.
L'adresse sera alors incrémentée de 1, afin de pointer sur le prochain caractère à afficher, puis appellera à nouveau la routine d'affichage.
Ce processus se déroulera jusqu'à affichage complet du texte, ce qui sous-entend que la longueur de notre texte soit connue au départ ; en effet, après chaque affichage de caractère, il faudra décrémenter cette longueur jusqu'à l'obtention de la valeur 0 ; afin que le système puisse décréter que l'affichage du texte est enfin terminé.

Vous voyez qu'en fait le système réalise un certain nombre d'actions pour arriver à ses fins...

Eh bien, en assembleur, c'est vous qui programmerez toutes ces actions : incrémentation du pointeur de caractères, appel de la routine d'affichage d'un caractère, décrémentation de la longueur du texte, et réitération du processus jusqu'à affichage du dernier caractère du texte.

Voyons comment ce programme assembleur va se présenter :

```
(1)   (2)      (3)   (4)      (5)            (6)

7000           10            ent # 7000     ; adr exec ass
7000           20            org # 7000     ; debut codes
7000  3E10     30            ld a,16        ; longueur = 16
7002  211070   40            ld hl, texte   ; init adr texte
7005  4F       50   affcar : ld c,a         ; sauve longueur
7006  7E       60            ld a, (hl)     ; car à afficher
7007  CD5ABB   70            call # bb5a    ; affichage car
700A  23       80            inc hl         ; +1 dans adr car
700B  79       90            ld a,c         ; recup longueur
700C  3D       100           dec a          ; -1 dans longueur
700D  20F6     110           jr nz, affcar  ; aller aff suivant
700F  C9       120           ret            ; fin aff texte

7010  54455854 130   texte : defm « TEXTE A AFFICHER »
```

Peut-être n'avez-vous pas compris en détail chaque instruction utilisée dans ce programme, mais nous allons tâcher d'y remédier dès à présent.

Vous remarquerez tout d'abord que le programme présenté comporte 6 colonnes, dont la numérotation apparaît en première ligne du programme.

Quand vous tapez les instructions assembleur, vous ne tapez en fait que les colonnes de 3 à 6.
Pour que votre programme soit exécutable, il faut qu'il soit transcrit en langage machine (voir colonne 2).
Pour ce faire, vous allez demander à votre assembleur d'effectuer ce travail fastidieux, mais ô combien nécessaire !
Si votre programme ne comporte aucune erreur, comme c'est le cas ici, vous obtiendrez les 6 colonnes présentées ici, sans aucun message d'erreur.

La première colonne indique l'adresse mémoire du premier code de l'instruction, une fois transcrite en codes hexadécimaux.
La deuxième colonne indique les codes hexa de l'instruction en question (voir colonne 5).
La troisième colonne indique un numéro séquentiel pour chacune des instructions du programme.

La sixième colonne permet d'indiquer un commentaire : ceci est en effet très utile pour documenter vos programmes en vue d'une utilisation ultérieure, ou pour une compréhension plus aisée par d'autres utilisateurs.

Si j'ai laissé pour la fin l'explication des colonnes 4 et 5, c'est que ce sont ces deux colonnes qui constituent le langage assembleur qui sera transcrit en langage machine (voir colonne 2).

La colonne 4 indique ce qu'on nomme en jargon assembleur un LABEL ou une ETIQUETTE ; pour faire la comparaison avec le langage Basic, nous pouvons dire qu'un GOTO 100, par exemple, équivaudrait ici à un JR NIVEAU, soit :

— en BASIC —	— en ASSEMBLEUR —
100 PRINT « SALUT ! » 200 GOTO 100	NIVEAU : CALL AFFICH JR NIVEAU

Passons maintenant à la colonne 5 qui est en fait le cœur de votre programme assembleur.

Nous pouvons déjà appréhender un certain nombre d'instructions. Nous distinguerons deux grands types d'instructions :

(a) les PSEUDO-OPERATIONS : telles que « ent », « org », etc.
(b) les instructions assembleur : telles que « ld c,a », « inc hl », etc.

LES PSEUDO-OPÉRATIONS

Ce sont des instructions qui ne sont pas transcrites en codes machine ; leur seul but est d'indiquer à l'assembleur certaines actions nécessaires pour le bon déroulement du programme.

Tous les assembleurs n'utilisent pas forcément les mêmes, et il faut être très vigilant quant à leur emploi, surtout si vous passez d'un assembleur à un autre.

Nous ne présenterons ici que les PSEUDO-OPERATIONS utilisées dans l'assembleur/désassembleur DEVPAC, puisque nous avons opté pour celui-ci afin de rédiger les programmes assembleur présentés dans cet ouvrage.

La pseudo-opération ENT

Cette instruction indique à l'assembleur l'adresse d'exécution du programme en mémoire ; on l'appelle « point d'entrée ».

ex. : ENT #7000 indique que le point d'entrée est en 7000 hexa.

La pseudo-opération ORG

ORG vient de l'anglais ORIGIN, qui signifie origine de l'implantation en mémoire. Cette instruction indique à l'assembleur l'adresse où sera logé le programme en mémoire ; en effet, l'adresse d'exécution n'est pas forcément l'adresse de la première instruction du programme.

ex. : ORG #6000 indique que votre programme commence à l'adresse hexa 6000

La pseudo-opération EQU

EQU vient de l'anglais EQUATE, qui signifie « équivalant à ». Cette instruction équivaut à donner une valeur entière à une variable ; en mathématiques, on dirait par exemple X = 345, eh bien, ici, on écrira :

X: EQU 345, ce qui permet d'attribuer la valeur 345 à la variable X.

La pseudo-opération DEFS

DEFS vient de l'anglais DEFINE STORAGE, qui signifie « définition d'une zone mémoire ».
Cette instruction permet d'allouer une zone mémoire, de N octets, initialisée à zéro, suivant l'endroit où est placée cette instruction.
Il est vivement recommandé de placer ce type d'instruction en fin de programme, afin d'éviter d'éventuelles surprises ; en effet, si vous voulez faire un programme relogeable, il est plus prudent de séparer instructions exécutables et instructions de définition de zones mémoire.
L'allocation d'une zone mémoire se fait en octet (8 bits).

ex. : ENTIER: DEFS 2 permet de définir une zone mémoire de 2 octets initialisée à 0, et qui s'appellera ENTIER.

La pseudo-opération DEFB

DEFB vient de l'anglais DEFINE BYTE, qui signifie « définition d'un octet », c'est-à-dire traitant une donnée sur 8 bits.
Cette instruction permet d'installer un ou plusieurs octets à partir de l'emplacement mémoire ou est située cette instruction. Comme pour l'instruction DEFS, il est vivement recommandé de la placer en fin de programme.
Vous pouvez installer des caractères ASCII (entre guillemets), ou tout simplement une valeur décimale ou hexadécimale ; le mélange de ces types de données ne pose aucun problème.

ex. : soit à installer MIMI + la valeur 0 (pour fin de texte).

Nous pouvons écrire par exemple l'instruction suivante :

```
                    TEXTE: DEFB "M","I",#4d,73,0
ou encore :         TEXTE: DEFB 40+37,"I",#4d,73,0
```

Bien entendu, il est plus simple d'écrire l'instruction suivante :

```
                    TEXTE: DEFB "M","I","M","I",0
ou encore :         TEXTE: DEFB 77,73,77,73,0
```

La pseudo-opération DEFW

DEFW vient de l'anglais DEFINE WORD, qui signifie « définition d'un mot binaire », c'est-à-dire traitant une donnée sur 16 bits. Cette instruction permet d'installer des données sur 2 octets, soit sur 16 bits ; **à cette différence près, elle est similaire à l'instruction DEFB**.

ex. : ADR: DEFW #95FF définit une adresse sur 2 octets.

La pseudo-opération DEFM

Cette instruction permet d'installer une chaîne de caractères ASCII (de 1 à 255 caractères).

```
ex. :               TEXTE: DEFM "MIMI"
ou encore :         TEXTE: DEFM "ABCDEFGHIJKL 12345678"
```

On remarquera que l'exemple utilisé pour DEFB aurait pu s'écrire :

```
TEXTE: DEFM "MIMI"
FINTX: DEFS 1
```

Il y a encore d'autres pseudo-opérations, mais leur emploi n'étant pas essentiel pour la suite, nous nous bornerons à celles-ci.

LES INSTRUCTIONS ASSEMBLEUR

Vous avez pu remarquer que certaines instructions permettent d'effectuer des transferts ou calculs portant sur des zones non définies en début du programme assembleur.

Si nous examinons l'instruction "LD A,16", nous voyons que nous avons affecté une longueur à une zone qui s'appelle "A", et qui est sur 1 octet (8 bits).

En effet, cette instruction aurait pu s'écrire en bon français :

METTRE LA VALEUR 16 DANS LA VARIABLE "A"

Par ailleurs, une instruction comme "INC HL" incrémente de 1 la zone qui s'appelle "HL", et qui est sur 2 octets (16 bits).

Mais n'allons pas plus loin, car ces zones ne sont pas des zones comme vous pouvez en définir en Basic ; en fait, ce sont des REGISTRES utilisés par votre ordinateur de façon privilégiée.

Ces registres représentent des positions mémoire qui font partie de la structure interne de votre microprocesseur.

Celui-ci dispose en effet d'un certain nombre de registres qui ne peuvent être utilisés qu'en assembleur.
Il existe des registres sur 8 bits et des registres, sur 16 bits. Toutefois, certains registres sur 8 bits peuvent se regrouper entre eux pour donner ce qu'on nomme communément en jargon assembleur des REGISTRES DOUBLES, et qui sont alors sur 16 bits.

LES REGISTRES DU Z80

Registres sur 8 bits

Registres : A / B / C / D / E / H / L

Seul le registre "a" est particulier, et se nomme ACCUMULATEUR. Ce registre est en effet une zone clé de la structure interne de votre microprocesseur ; c'est par celui-ci que passe la majorité des opérations dites ARITHMETIQUES ou LOGIQUES.
Il offre également une rapidité d'exécution au niveau des E/S. De plus, il est certainement le registre le plus utilisé.

Registres sur 16 bits

Registres : IX / IY

Ces deux registres sont appelés REGISTRES D'INDEX ; ils ont pour but de permettre l'adressage de façon indexée, ou si vous voulez, suivant le principe d'un indice.

Registre : SP

SP, de l'anglais STACK POINTER, signifie « pointeur de pile ». Ce registre est utilisé pour fixer l'adresse d'une zone mémoire, que l'on appelle "pile" ; il s'agit en fait du POINTEUR DE PILE. Ce registre est toutefois peu utilisé dans les programmes, car il est fort rare d'avoir besoin de sauver plus de 256 octets, et de plus la gestion en serait assez complexe.

Les registres doubles

En fait il n'existe pas de vrais registres doubles, puisque ceux-ci sont obtenus à partir des registres sur 8 bits, à l'exclusion du registre "a".

Registres doubles : BC / DE / HL

Ce sont en effet les seules combinaisons offertes par le Z80. Ces registres doubles permettent d'adresser des positions mémoire sur 16 bits, soit de 0 à #FFFF (65535).

Toutefois, pour une utilisation optimale de ces doubles registres, il est préférable d'utiliser la paire de registres "HL" pour des adressages mémoire ; en effet, le temps d'exécution est nettement plus rapide qu'avec les deux autres paires de registres "BC" et "DE".

En général, les paires de registres "BC" et "DE" sont utilisées comme des mémoires-TAMPONS, soit pour des adresses, soit pour des données sur 16 bits.

Nous allons parler maintenant d'un registre un peu particulier, qui se nomme le REGISTRE D'ETAT.
Ce registre est associé au registre "a" pour former un registre double qui s'appelle "af".
Sa fonction est des plus importantes, car c'est grâce à lui que nous pouvons connaître le résultat de toute opération effectuée par n'importe quelle instruction assembleur. Pour connaître le résultat de l'exécution d'une instruction, il faut aller voir tel ou tel bit de ce registre, afin d'effectuer un traitement en conséquence.

FONCTIONS DES BITS DU REGISTRE D'ÉTAT

BIT 0 : il s'agit de l'indicateur de retenue (appelé en jargon assembleur CARRY) ; il est affecté par les instructions arithmétiques, logiques, et décalages.

BIT 1 : il s'agit de l'indicateur de soustraction, mis à 0 par les instructions d'addition, et à 1 par les instructions de soustraction.

BIT 2 : il s'agit de l'indicateur de dépassement (ou overflow), dans le cas d'opérations arithmétiques.
Il exerce également une détection de la parité pour certaines instructions de rotation.

BIT 3 : ce bit n'est pas utilisé.

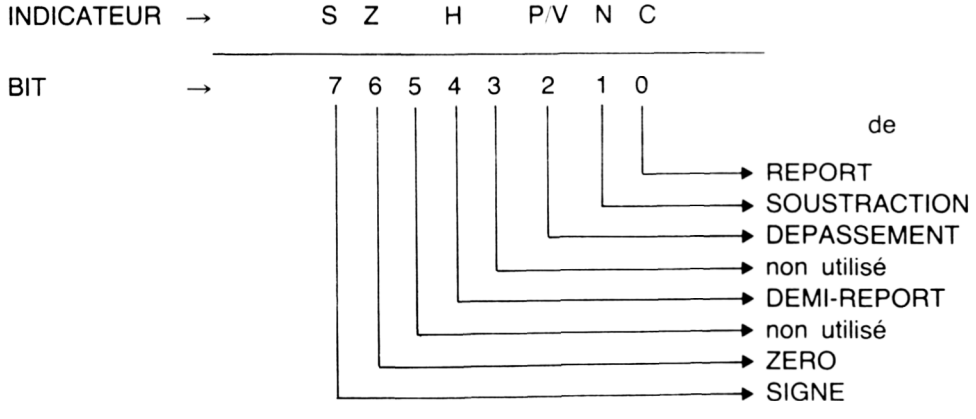
BIT 4 : il s'agit de l'indicateur de demi-retenue ; il a la même fonction que le bit 0, mais pour des opérations sur un octet.

BIT 5 : ce bit n'est pas utilisé.

BIT 6 : il s'agit de l'indicateur de zéro ; si le résultat d'une opération donne un résultat nul, il est mis à 1, sinon il est mis à 0.

BIT 7 : il s'agit de l'indicateur de signe ; il est mis à 1 si le bit le plus significatif d'une opération arithmétique ou logique est 1, sinon il est mis à 0.

En résumé, le registre d'état (ou registre "f") a la structure suivante :



TEST DES VALEURS PRISES PAR LES INDICATEURS

Nous avons vu dans les deux programmes assembleur présentés jusqu'à présent qu'une instruction "JR NZ, étiquette" permettait de se brancher à "étiquette" si le résultat de la précédente instruction était différent de 0 ; sinon on continuait en séquence si ce résultat était nul

Nous voyons que l'abrévation employée est "NZ", c'est-à-dire, si le résultat est non nul, aller à "étiquette".

Nous avons donc le choix entre un certain nombre d'abrévations que nous vous présentons ci-après :

abrévation	valeur	bit	Signification
C	1	0	trait. ayant entraîné un report
NC	0	0	traitement sans report
Z	1	6	trait. dont résultat nul
NZ	0	6	trait. dont résultat non nul
PO	1	2	trait. dont parité paire
PE	0	2	trait. dont parité impaire
M	1	7	trait. dont résultat négatif
P	0	7	trait. dont résultat positif

LES INSTRUCTIONS DU Z80

Après ces préliminaires indispensables, passons aux instructions assembleur proprement dites :
Nous allons passer en revue la majorité des instructions du Z80, et tenter de nous imprégner de leur mécanisme fondamental.

LEGENDES UTILISEES : d8 = donnée 8 bits / r = registre 8 bits
dd = donnée ou adr. / rr = registre double (sur 16 bits)
e = déplacement relatif (1 octet) pour JR

Instructions d'addition avec retenue → ADC

Ce type d'instruction permet d'additionner la retenue à deux éléments, ces éléments pouvant être un registre simple ou double, une donnée immédiate (ex. : 125), ou une donnée en mémoire.

instructions	fonctions
ADC A,r	addition de r, de A, et du report, dans A.
ADC A,d8	addition de d8 et de A dans A.
ADC A,(HL)	addition d'une donnée pointée par HL à A et du report dans A.
ADC HL,rr	addition de rr de HL et du report dans HL.
ADC HL,SP	addition de SP de HL et du report dans HL.
ADC A,(IX+d8)	addition du report de A et d'une donnée pointée par IX+d8.
ADC A, (IY+d8)	idem que ci-dessus mais avec IY+d8.

Instructions d'addition sans report → ADD

Nous retrouvons les instructions similaires à ADC, et en plus :

instructions	fonctions
ADD IX,rr	addition de BC ou DE avec IX dans IX
ADD IX,SP	addition de SP avec IX dans IX
ADD IY,rr	addition de BC ou DE avec IY dans IY
ADD IY,SP	addition de SP avec IY dans IY

Instructions effectuant un ET logique → AND

rappel : un ET logique est une fonction booléenne qui donne les résultats suivants :

$0 \times 0 = 0$
$0 \times 1 = 0$ (commutatif)
$1 \times 0 = 0$ (commutatif)
$1 \times 1 = 1$

ex. : AND 0 effectue un ET logique entre l'accumulateur et la donnée immédiate 0 ; ceci aura pour effet de mettre systématiquement l'accumulateur à 0, ce qui peut être très utile...

Les possibilités sont les suivantes :

instructions	fonctions
AND r	effectue un ET logique entre r et A dans A
AND d8	effectue un ET logique entre d8 et A dans A
AND (HL)	ET logique entre une donnée pointée par HL et A dans A.
AND (IX+d8)	ET logique entre une donnée pointée par IX+d8 et A dans A.
AND (IY+d8)	idem ci-dessus mais avec IY+d8

Instructions de test d'un bit donné → BIT

Les instructions BIT sont de la forme : BIT b, opérande "b" pouvant prendre les valeurs 0 à 7 (pour un octet).

L'opérande peut être, soit un registre, soit (HL), soit (IX+d8), soit (IY+d8).

ex. : BIT 0,(HL) . permet de tester le bit 0 de la donnée pointée par l'adresse contenue dans HL.

BIT 1,(IX+d8) permet de tester le bit 1 de la donnée pointée par le contenu du registre index IX + d8.

Instructions d'appel de sous-programme → CALL

Il est possible d'exécuter un sous-programme avec ou sans condition particulière. L'équivalent Basic serait un GOSUB n, ou un ON × GOSUB n1,n2,n3.

Les possibilités sont les suivantes :

instructions	fonctions
CALL dd	appel d'un s–p suivant l'adresse dd
CALL Z,dd	appel d'un s–p d'adr. dd suiv. condition Z
CALL NZ,dd	appel d'un s–p d'adr. dd suiv. condition NZ
CALL C,dd	appel d'un s–p d'adr. dd suiv. condition C
CALL NC,dd	appel d'un s–p d'adr. dd suiv. condition NC
CALL PO,dd	appel d'un s–p d'adr. dd suiv. condition PO
CALL PE,dd	appel d'un s–p d'adr. dd suiv. condition PE
CALL M,dd	appel d'un s–p d'adr. dd suiv. condition M
CALL P,dd	appel d'un s–p d'adr. dd suiv. condition P

Instructions de comparaison → CP

Ce type d'instruction permet de comparer un opérande avec le registre A, en effectuant la soustraction (A – opérande), mais le résultat de cette soustraction n'est pas conservé. A demeure inchangé.

Les possibilités sont les suivantes :

instructions	fonctions
CP r	comparaison de A et du registre r
CP d8	comparaison de A et de la donnée d8
CP (HL)	comp. de A et de la donnée pointée par HL
CP (IX+d8)	comp. de A et de la donnée pointée par IX+d8
CP (IY+d8)	comp. de A et de la donnée pointée par IY+d8

Il existe d'autres instructions de comparaison, mais qui sont bien moins utilisées : CPD / CPDR / CPI / CPIR.
Ces instructions permettent d'effectuer des comparaisons sur des blocs de données, avec incrémentation ou décrémentation automatique des pointeurs utilisés (BC et HL).

Instructions de décrémentation → DEC

Ce type d'instruction permet de décrémenter de 1 un opérande spécifié.

Les possibilités sont les suivantes :

instructions	fonctions
DEC r	décrémentation du registre r
DEC rr	décrémentation du registre double rr
DEC IX	décrémentation du registre IX
DEC IY	décrémentation du registre IY
DEC SP	décrémentation du pointeur de pile
DEC (HL)	décrémentation de la donnée pointée par HL
DEC (IX+d8)	décrémentation de la donnée pointée par IX+d8
DEC (IY+d8)	décrémentation de la donnée pointée par IY+d8

Instructions d'échange des registres → EX

Avant de parler des différentes possibilités d'échange entre les registres, il nous faut compléter nos connaissances sur l'existence des registres du Z80.
En effet, il existe des REGISTRES AUXILIAIRES, qui sont en fait des copies des registres AF, BC, DE, HL ; ces registres se nomment AF′, BC′, DE′, et HL′.
Ils sont généralement utilisés pour les interruptions ; ils sont également utiles pour effectuer des sauvegardes des registres courants.

Les possibilités sont les suivantes :

instructions	fonctions
EX AF,AF′	échange A et reg. état avec son auxiliaire
EX DE,HL	échange des reg. doubles HL et DE
EX (SP),HL	échange de HL avec le pointeur de pile
EX (SP),IX	échange de IX avec le pointeur de pile
EX (SP),IY	échange de IY avec le pointeur de pile
EXX	échange BC avec BC′, HL avec HL′, DE avec DE′

Instructions d'incrémentation → INC

Ce type d'instruction permet d'incrémenter de 1 un opérande spécifié.

Les possibilités sont les suivantes :

instructions	fonctions
INC r	incrémentation du registre r
INC rr	incrémentation du registre double rr
INC IX	incrémentation du registre IX
INC IY	incrémentation du registre IY
INC SP	incrémentation du pointeur de pile
INC (HL)	incrémentation de la donnée pointée par HL
INC (IX+d8)	incrémentation de la donnée pointée par IX+d8
INC (IY+d8)	incrémentation de la donnée pointée par IY+d8

Instructions de saut conditionnel ou non → JP

Il est possible de se brancher à une adresse, avec ou sans condition particulière. L'équivalent Basic serait un GOTO n, ou ON × GOTO n1,n2,n3.

Les possibilités sont les suivantes :

instructions	fonctions
JP dd	saut à l'adresse dd
JP NZ,dd	saut à l'adresse dd suivant la condition NZ
JP Z,dd	saut à l'adresse dd suivant la condition Z
JP NC,dd	saut à l'adresse dd suivant la condition NC
JP N,dd	saut à l'adresse dd suivant la condition N
JP PE,dd	saut à l'adresse dd suivant la condition PE
JP PO,dd	saut à l'adresse dd suivant la condition PO
JP M,dd	saut à l'adresse dd suivant la condition M
JP P,dd	saut à l'adresse dd suivant la condition P
JP (HL)	saut à l'adresse suivant le contenu de HL
JP (IX)	saut à l'adresse suivant le contenu de IX
JP (IY)	saut à l'adresse suivant le contenu de IY

Instructions de saut relatif, avec ou sans condition → JR

L'instruction JR permet d'effectuer un saut maximum de 126 octets en arrière, ou un saut maximum de 129 octets en avant ; le saut est dit relatif, car celui-ci est calculé à partir du dernier octet de l'instruction JR.

Elle est en général plus utilisée qu'un JP pour 2 raisons :

(a) la majorité des sauts excède rarement les valeurs citées ci-dessus (soit de −126 à +129).

(b) cette instruction n'utilise que 2 octets, alors qu'un JP en utilise 3 (à cause de l'adressage absolu).

Les possibilités sont les suivantes :

instructions	fonctions
JR e	saut à l'adresse suivant la valeur e
JR NZ,e	saut suivant e, et conditionné par NZ
JR Z,e	saut suivant e, et conditionné par Z
JR NC,e	saut suivant e, et conditionné par NC
JR C,e	saut suivant e, et conditionné par C

Instructions de chargement → LD

Le Z80 offre un nombre assez impressionnant de possibilités de chargement.

Afin d'avoir une idée assez précise de ce qui est proposé, nous allons présenter un tableau scindé en 8 groupes.
Ces groupes ont pour objectif de montrer les différents types de chargement qui ont été envisagés pour le Z80, à savoir :

Charger un registre / charger le registre A / charger un registre double / charger une zone mémoire / charger les registres index / charger le pointeur de pile / charger une zone mémoire pointée par un registre double / charger une zone mémoire pointée par un registre index incrémenté d'une donnée sur 8 bits.

Les instructions faisant appel aux registres d'interruption et de rafraîchissement ne seront pas présentées, car elles sont peu utilisées par le programmeur amateur.

instructions	fonctions
LD r,r	charger registre par registre
LD r,d8	charger registre par donnée 8 bits
LD r,(IX+d8)	charger registre par donnée pointée par IX+d8
LD r,(IY+d8)	charger registre par donnée pointée par IY+d8
LD r,(HL)	charger registre par donnée pointée par HL
LD A,(dd)	charger A par donnée pointée par dd
LD A,(BC)	charger A par donnée pointée par BC
LD A,(DE)	charger A par donnée pointée par DE
LD rr,(dd)	charger reg. double par donnée pointée par dd
LD rr,dd	charger reg. double par donnée dd
LD (dd),A	charger zone dd par A
LD (dd),rr	charger zone dd par registre double
LD (dd),IX	charger zone dd par registre IX
LD (dd),IY	charger zone dd par registre IY
LD IX,dd	charger registre IX par dd
LD IX,(dd)	charger registre IX par donnée pointée par dd
LD IY,dd	charger registre IY par dd
LD IY,(dd)	charger registre IY par donnée pointée par dd
LD (IX+d8),d8	charger zone pointée par IX+d8 par d8
LD (IX+d8),r	charger zone pointée par IX+d8 par registre
LD (IY+d8),d8	charger zone pointée par IY+d8 par d8
LD (IY+d8),r	charger zone pointée par IY+d8 par registre
LD SP,HL	charger pointeur de pile par HL
LD SP,IX	charger pointeur de pile par IX
LD SP,IY	charger pointeur de pile par IY
LD (DE),A	charger donnée pointée par DE par A
LD (BC),A	charger donnée pointée par BC par A
LD (HL),r	charger donnée pointée par HL par registre
LD (HL),d8	charger donnée pointée par HL par d8

Instructions de transfert de bloc → LDx

Ce type d'instruction permet de transférer un ou plusieurs blocs de données d'un octet.

Les possibilités sont les suivantes :

instructions	fonctions
LDD	transf. 1 octet pointé par HL dans une zone pointée par DE, puis BC/DE/HL sont decr. de 1
LDDR	transf. n octets suivant BC, et pointés par HL, dans une zone pointée par DE. BC/DE/HL sont décrémentés de 1 jusqu'à BC=0 (fin transfert)
LDI	transf. 1 octet pointé par HL dans une zone pointée par DE, puis BC/DE/HL sont incr. de 1
LDIR	transf. n octets suivant BC, et pointés par HL, dans une zone pointée par DE. BC/DE/HL sont incrémentés de 1 jusqu'à BC=0 (fin transfert)

Instructions effectuant un OU inclusif → OR

rappel : un OU inclusif est une fonction booléenne qui donne les résultats suivants :

0 + 0 = 0
0 + 1 = 1 (commutatif)
1 + 0 = 1 (commutatif)
1 + 1 = 1

ex. : OR #FF effectue un OU inclusif entre A et #FF ; ceci aura pour effet de mettre systématiquement l'accumulateur à #FF, tout en n'utilisant qu'un octet pour cela.

autre exemple : OR #9C (avec A = #A7)

le résultat obtenu est #BF

Les possibilités sont les suivantes :

instructions	fonctions
OR r	effectue un OU inclus. entre r et A dans A
OR d8	effectue un OU inclus. entre d8 et A dans A
OR (HL)	OU inclus. entre une donnée pointée par HL et A dans A.
OR (IX+d8)	OU inclus. entre une donnée pointée par IX+d8 et A dans A.
OR (IY+d8)	idem ci-dessus mais avec IY+d8

Instructions de traitement sur la pile → PUSH / POF

(a) l'instructions PUSH permet de charger le contenu d'un registre 16 bits en haut de la pile (empilement).
Elle permet de sauvegarder le registre 16 bits traité.

Les possibilités sont : PUSH AF / PUSH BC / PUSH DE
PUSH HL / PUSH IX / PUSH IY

(b) l'instruction POP permet de charger un registre 16 bits à partir du haut de la pile (dépilement).
Elle permet de récupérer un registre 16 bits sauvegardé par un PUSH, mais pas forcément dans ce même registre 16 bits.

Les possibilités sont : POP AF / POP BC / POP DE
POP HL / POP IX / POP IY

exemple :

(voir)
pile AVANT routine

```
: PUSH AF
: PUSH BC
: PUSH DE
: PUSH HL
  CALL routine
→ POP HL
→ POP DE
→ POP BC
→ POP AF
```

(voir)
pile APRES récup.

PILE

AVANT	APRES	
arc	dd1	Haut
HL	dd2	
DE	dd3	
BC	dd4	
AF	dd5	
dd1	dd6	Bas

Nota : "arc", dans la pile, correspond à l'adresse de retour du CALL ; celle-ci est gérée par le système ; toutefois si vous devez effectuer des PUSH ou POP dans la routine appelée, n'oubliez pas que cette adresse est en haut de la pile...
La routine appelée modifiant les registres AF/BC/DE/HL, il est nécessaire de procéder à une sauvegarde de ceux-ci pour les récupérer après exécution de la routine.
Noter au passage l'importance de la symétrie pour la récupération des registres sauvés, en raison du système de gestion de la pile.

Le principe d'empilement et de dépilement de la pile se fait suivant la méthode LIFO ; ce terme, tiré de l'anglais Last In First Out, signifie « dernier entré Premier sorti », d'où la symétrie évoquée précédemment.

Instructions de mise à 0 d'un bit → RES

Les instructions RES sont de la forme : RES b, opérande "b" pouvant prendre les valeurs 0 à 7 (pour un octet).

L'opérande peut être soit un registre, soit (HL), soit (IX+d8), soit (IY+d8).

ex. : RES 2,(HL) permet de mettre à 0 le bit 2 de la donnée pointée par l'adresse contenue dans HL.

RES 7,(IY+d8) permet de mettre à 0 le bit 7 de la donnée pointée par le contenu de l'index IY + d8.

Instructions de retour de S/P avec ou sans condition → RET

Nous avons vu plus haut qu'il était possible d'appeler un S/P par un CALL (équivalent d'un GOSUB en BASIC) ; comme vous le savez aussi, le retour d'un S/P BASIC se fait par un RETURN.

Eh bien, pour un S/P assembleur, il en va de même, mais avec une instruction RET (comme RETurn).

Les possibilités sont les suivantes :

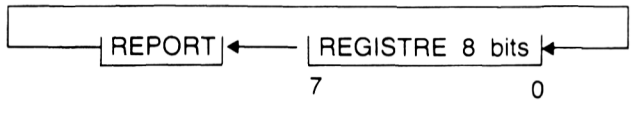
instructions	fonctions
RET	retour d'un Sous-Programme
RET Z	retour d'un S/P suivant la condition Z
RET NZ	retour d'un S/P suivant la condition NZ
RET C	retour d'un S/P suivant la condition C
RET NC	retour d'un S/P suivant la condition NC
RET PO	retour d'un S/P suivant la condition PO
RET PE	retour d'un S/P suivant la condition PE
RET M	retour d'un S/P suivant la condition M
RET P	retour d'un S/P suivant la condition P

Instruction de rotation à gauche à travers le report → RL

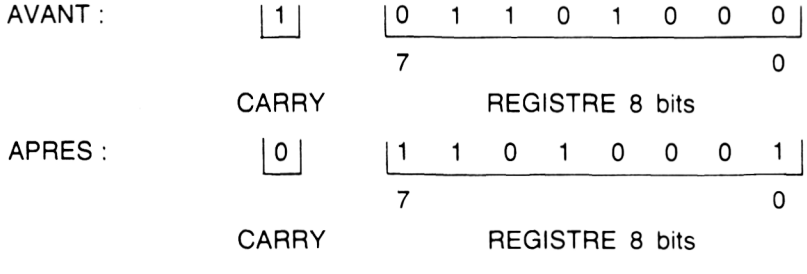
Dans ce type d'instruction, il s'agit de faire passer, pour un registre ou une zone sur 8 bits, chacun des bits donnés dans le bit suivant ; soit le bit 0 dans le bit 1, le bit 1 dans le bit 2, etc.

Le bit 7 quant à lui passe dans le REPORT, et le REPORT passe dans le bit 0.

Schématiquement, cela nous donne :



exemple de rotation à gauche :



Les possibilités sont les suivantes :

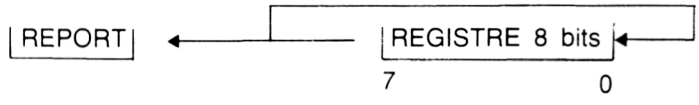
instructions	fonctions
RL r	rotation d'un registre
RL (HL)	rotation d'une donnée pointée par HL
RL (IX+d8)	rotation d'une donnée pointée par IX+d8
RL (IY+d8)	rotation d'une donnée pointée par IY+d8
RLA	rotation du reg. A (1 oct. de moins que RL A)

Instruction de rotation à gauche sans le report → RLC

Ce type d'instruction est similaire à l'instruction RL, à la différence que le bit 7 va dans le report et le bit 0.

Le report quant à lui ne sert qu'à réceptionner le contenu du septième bit de la donnée ou registre sur 8 bits.

Schématiquement, cela nous donne :



exemple de rotation à gauche :

Les possibilités sont les suivantes :

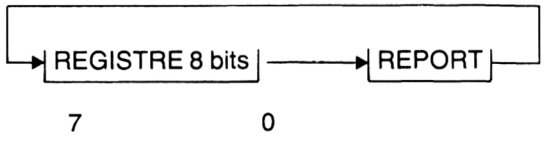
Instructions	Fonctions
RLC r	rotation d'un registre
RLC (HL)	rotation d'une donnée pointée par HL
RLC (IX+d8)	rotation d'une donnée pointée par IX+d8
RLC (IY+d8)	rotation d'une donnée pointée par IY+d8
RLCA	rotation de A (1 oct. de moins que RLC A)

Instruction de rotation à droite à travers le report → RR

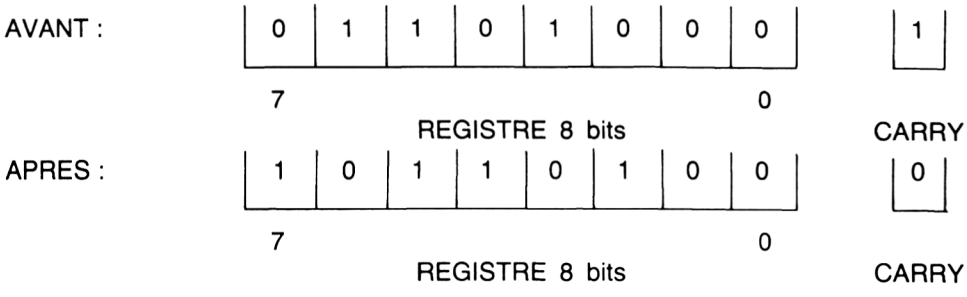
Dans ce type d'instruction, il s'agit de faire passer, pour un registre ou une zone sur 8 bits, chacun des bits donnés dans le bit précédent ; soit le bit 7 dans le bit 6, le bit 6 dans le bit 5, etc.

Le bit 0 quant à lui passe dans le REPORT, et le REPORT passe dans le bit 7.

Schématiquement, cela nous donne :



exemple de rotation à droite :



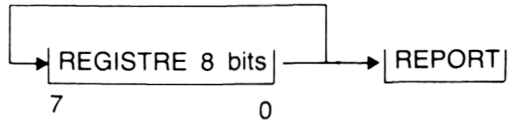
Les possibilités sont les suivantes :

Instructions	Fonctions
RR r	rotation d'un registre
RR (HL)	rotation d'une donnée pointée par HL
RR (IX+d8)	..... IX+d8
RR (IY+d8)	..... IY+d8
RRA	rotation du reg. A (1 oct. de moins que RR A)

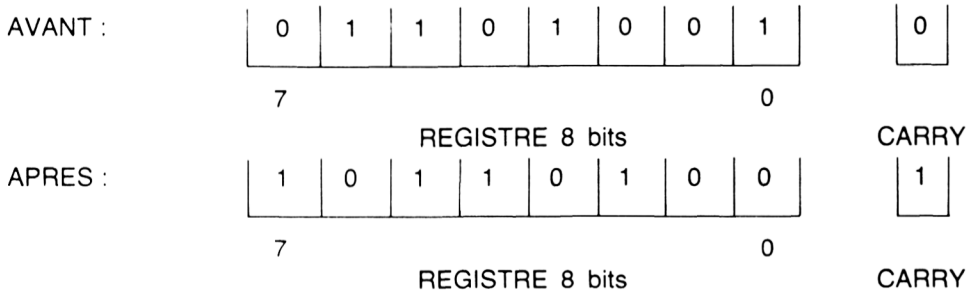
Instruction de rotation à droite sans le report → RRC

Ce type d'instruction est similaire à l'instruction RR, à la différence que le bit 0 va dans le report et le bit 7
Le report quant à lui ne sert qu'à réceptionner le contenu du bit 0 de la donnée ou registre sur 8 bits.

Schématiquement, cela nous donne :



exemple de rotation à gauche :



Les possibilités sont les suivantes :

Instructions	Fonctions
RRC r	rotation d'un registre
RRC (HL)	rotation d'une donnée pointée par HL
RRC (IX+d8)	..... IX+d8
RRC (IY+d8)	..... IY+d8
RRCA	rotation de A (1 oct. de moins que RRC A)

Instructions restarts → RST

Ce type d'instruction (sur un octet) permet d'effectuer un saut dans les premières adresses de la page 0 de votre micro équipé d'un Z80 ; 7 restarts sont fournis, et seront numérotés de 0 à 7.

Les possibilités sont les suivantes :

Instructions	Fonctions	numéro RST
RST 0	effectue un saut à l'adresse 0	0
RST 8	effectue un saut à l'adresse 8	1
RST #10	effectue un saut à l'adresse 16	2
RST #18	effectue un saut à l'adresse 24	3
RST #20	effectue un saut à l'adresse 32	4
RST #28	effectue un saut à l'adresse 40	5
RST #30	effectue un saut à l'adresse 48	6
RST #38	effectue un saut à l'adresse 56	7

En ce qui concerne l'Amstrad, seul le RST #30 est réservé à l'utilisateur ; il peut être utilisé pour l'exécution d'un S/P dont l'appel est très fréquent ; l'avantage réside dans le fait que vous n'avez besoin que d'un seul octet, alors qu'une instruction CALL en nécessite trois.

Voyons sommairement à quoi peuvent bien servir les autres RST.

RST Num.	Fonctions
0	— effectue un RESET (équivalent du SHIFT/CTRL/ESC)
1	— effectue un saut en ROM ou RAM basse, et demande pour cela 2 octets à la suite du code RST.
2	— effectue un saut dans une ROM d'extension, et requiert aussi 2 octets à la suite du code RST.
3	— effectue un saut en RAM ou en ROM, et nécessite pour cela 2 octets à la suite du code RST.
4	— effectue un saut en RAM, et requiert pour cela 2 octets à la suite du code RST.
5	— effectue un saut en ROM basse, et réclame pour ce faire 2 octets à la suite du code RST.
7	— ce RST permet de gérer toute interruption (1 oct.)

Instructions de soustraction → SBC

Ce type d'instruction permet d'effectuer une soustraction, de l'accumulateur ou d'un registre 16 bits, d'un opérande et du report.

ex. : SBC A,B effectue l'opération A = A − B − REPORT

Il est donc important de s'assurer de la valeur du report avant de procéder à l'exécution d'une telle instruction.

Les possibilités sont les suivantes :

instructions	fonctions
SBC A,r	soustrait de A un registre et le report
SBC A,d8	soustrait de A d8 et le report
SBC A,(HL)	soustrait de A une donnée pointée par HL et le report
SBC A,(IX+d8)	soustrait de A une donnée pointée par IX+d8 et le report
SBC A,(IY+d8)	soustrait de A une donnée pointée par IY+d8 et le report
SBC HL,rr	soustrait de HL le reg. double rr et le carry
SBC HL,SP	soustrait de HL le point. de pile et le carry

Instructions de mise à 1 d'un bit → SET

Les instructions SET sont de la forme : SET b, opérande "b" pouvant prendre les valeurs 0 à 7 (pour un octet).

L'opérande peut être soit un registre, soit (HL), soit (IX+d8), ou encore (IY+d8).

ex. : SET 0,(HL) permet de mettre à 1 le bit 0 de la donnée pointée par l'adresse contenue dans HL.

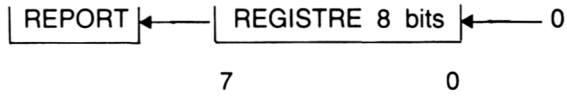
SET 3,(IX+d8) permet de mettre à 1 le bit 3 de la donnée pointée par le contenu de l'index IX + d8.

Instructions de décalage à gauche d'un opérande → SLA

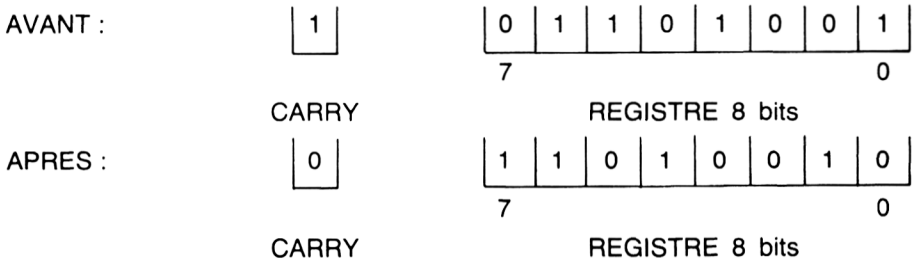
Dans ce type d'instruction, il s'agit de faire passer, pour un registre ou une zone sur 8 bits, chacun des bits donnés dans le bit suivant ; soit le bit 0 dans le bit 1, le bit 1 dans le bit 2, etc.

Le bit 7 quant à lui passe dans le REPORT, et une valeur 0 passe dans le bit 0.

Schématiquement, cela nous donne :



exemple de décalage à gauche :



Les possibilités sont les suivantes :

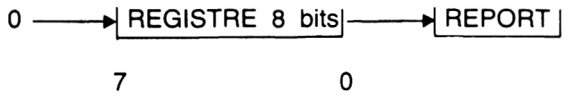
instructions	fonctions
SLA r	décalage d'un registre
SLA (HL)	décalage d'une donnée pointée par HL
SLA (IX+d8)	décalage d'une donnée pointée par IX+d8
SLA (IY+d8)	décalage d'une donnée pointée par IY+d8

Instructions de décalage à droite d'un opérande → SRL

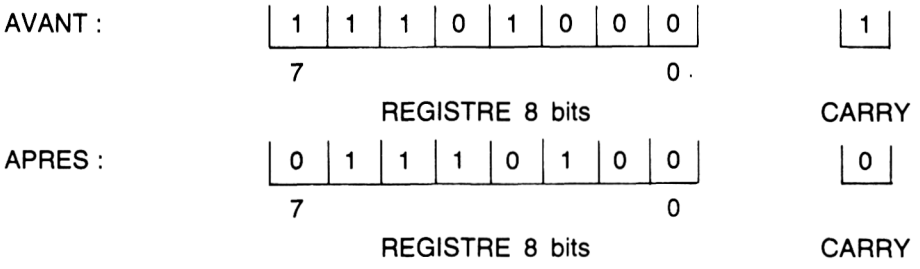
Dans ce type d'instruction, il s'agit de faire passer, pour un registre ou une zone sur 8 bits, chacun des bits donnés dans le bit précédent ; soit le bit 7 dans le bit 6, le bit 6 dans le bit 5, etc.

Le bit 0 quant à lui passe dans le REPORT, et une valeur 0 passe dans le bit 7.

Schématiquement, cela nous donne :



exemple de décalage à droite :



Les possibilités sont les suivantes :

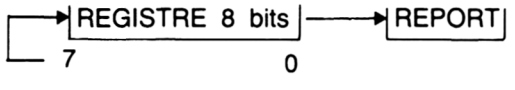
instructions	fonctions
SLR r	décalage d'un registre
SRL (HL)	décalage d'une donnée pointée par HL
SRL (IX+d8)	décalage d'une donnée pointée par IX+d8
SRL (IY+d8)	décalage d'une donnée pointée par IY+d8

Instructions de décalage à droite d'un opérande → SRA

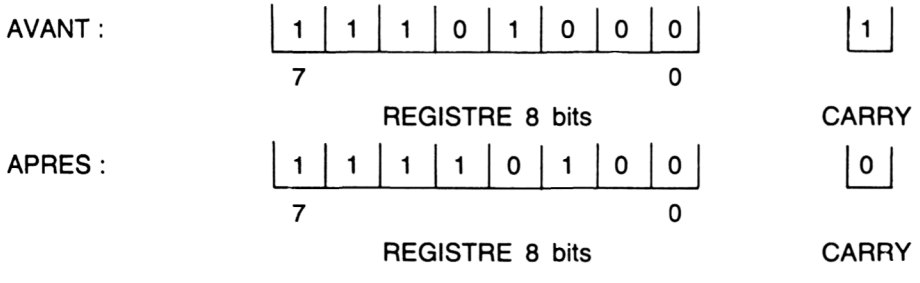
Dans ce type d'instruction, il s'agit de faire passer, pour un registre ou une zone sur 8 bits, chacun des bits donnés dans le bit précédent ; soit le bit 7 dans le bit 6, le bit 6 dans le bit 5, etc.

Le bit 0 quant à lui passe dans le REPORT, et le bit 7 reste inchangé.

Schématiquement, cela nous donne :



exemple de décalage à droite :



Les possibilités sont les suivantes :

instructions	fonctions
SRA r	décalage d'un registre
SRA (HL)	décalage d'une donnée pointée par HL
SRA (IX+d8)	décalage d'une donnée pointée par IX+d8
SRA (IY+d8)	décalage d'une donnée pointée par IY+d8

Instructions de soustraction de l'accumulateur → SUB

Ce type d'instruction permet de soustraire une opérande de A.

Les possibilités sont les suivantes :

instructions	fonctions
SUB r	soustrait de A un registre r
SUB d8	soustrait de A une donnée d8
SUB (HL)	soustrait de A une donnée pointée par HL
SUB (IX+d8)	soustrait de A une donnée pointée par IX+d8
SUB (IY+d8)	soustrait de A une donnée pointée par IY+d8

Instructions effectuant un OU exclusif → XOR

Rappel : un OU exclusif est une fonction booléenne qui donne les résultats suivants :

0 + 0 = 0
0 + 1 = 1 (commutatif)
1 + 0 = 1 (commutatif)
1 + 1 = 0

ex. : XOR A effectue un OU exclusif entre A et lui-même ; ceci aura pour effet de mettre systématiquement l'accumulateur à 0. Le XOR A et le AND 0 ont le même effet, et n'utilisent pour ce faire qu'un seul octet.

Les possibilités sont les suivantes :

instructions	fonctions
XOR r	effectue un OU exclus. Entre r et A dans A
XOR d8	effectue un OU exclus. Entre d8 et A dans A
XOR (HL)	Ou exclus. entre une donnée pointée par HL et A dans A.
XOR (IX+d8)	OU exclus. entre une donnée pointée par IX+d8 et A dans A
XOR (IY+d8)	idem ci-dessus mais avec IY+d8

Instructions diverses (ni interruptions, ni E/S)

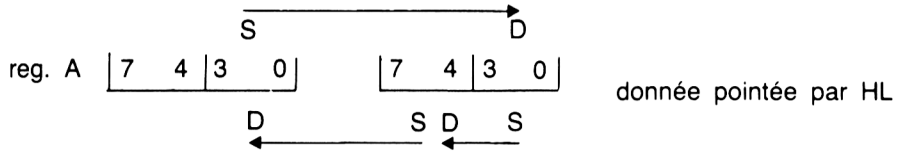
instructions	fonctions
CCF	complémentation de l'indicateur de report
CPL	complémentation de l'accumulateur
DAA	mise de A sous forme DCB (décimal codé bin.) à utiliser après : ADC/ADD/INC/SBC/SUB/DEC/NEG.
DJNZ e	décrémente B et saut relatif tant que B <> 0
HALT	suspend l'exécution de l'unité centrale
NEG	négation de A, soit A = 0 − A
NOP	instruction nulle
SCF	positionne l'indicateur de report à 1

Instructions de rotation décimale : RLD / RRD

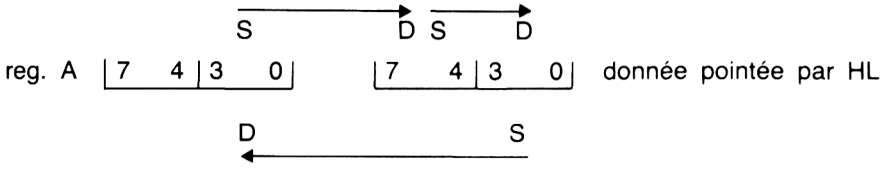
Pour ces 2 instructions, les transferts se font par quartet, c'est-à-dire par groupe de 4 bits ; le quartet droit correspond aux bits 0 à 3, et le quartet gauche correspond aux bits 4 à 7.

→ S représente le quartet à transférer (source).
→ D représente le quartet de réception (destination).

(a) rotation décimale à gauche : RLD



(b) rotation décimale à droite : RRD



Instructions d'interruption et rafraîchissement

Nous n'avons en effet pas évoqué deux autres registres qui sont :

le registre d'interruption I
le registre de rafraîchissement R

Les instructions utilisant ces deux registres demandant une très bonne connaissance du système de gestion des interruptions, nous ne ferons que les citer ; je vous incite à lire des ouvrages très spécialisés pour en savoir plus sur ce sujet.

instructions	fonctions
DI	interdiction des interruptions
EI	autorisation des interruptions
IM 0	sélection du mode d'interruption 0
IM 1	sélection du mode d'interruption 1
IM 2	sélection du mode d'interruption 2
LD A,I	charger I dans A
LD I,A	charger A dans I
LD A,R	charger R dans A
LD R,A	charger A dans R
RETI	retour d'interruption
RETN	retour d'interruption non masquable

Instructions d'entrée-sortie (E/S) → IN / OUT

Il faut tout d'abord savoir qu'un périphérique peut être ou lu ou écrit par l'intermédiaire d'une zone qui se nomme PORT de données.

Le port de données est une zone de 8 bits propre à chaque E/S.

Instructions de lecture d'un périphérique par le port de données :

instructions	fonctions
IN r,(C)	— charger r à partir du port pointé par C
IN A,(d8)	— charger A à partir du port d8
IND	— charger zone pointée par HL par le contenu du port pointé par C. B et HL sont ensuite décr.
INDR	— même fonction que IND mais répétée jusqu'à ce que B=0 ; permet de lire jusqu'à 255 octets ; lecture par décr. du pointeur de zone : HL
INI	— charger zone pointée par HL par le contenu du port pointé par C. puis −1 dans B, +1 dans HL
INIR	— même fonction que INI mais répétée jusqu'à ce que B=0 ; permet de lire jusqu'à 255 octets ; lecture par incr. du pointeur de zone : HL

Instructions d'écriture d'un périphérique par le port de données.

instructions	fonctions
OUT (C),r	fonction inverse de IN r,(C), soit écriture
OUT (d8),A	fonction inverse de IN A,(d8), soit écriture
OUTD	fonction inverse de IND, soit écriture
OUTI	fonction inverse de INI, soit écriture
OTDR	fonction inverse de INDR, soit écriture
OTIR	fonction inverse de INIR, soit écriture

MODIFICATION DES INDICATEURS DU REGISTRE D'ÉTAT

LÉGENDES UTILISÉES

dans INSTRUCTIONS :
op = opérande / r = registre simple / rr = registre double
dans INDICATEURS :
M = modification de l'indicateur suivant le résultat obtenu
P = modification de l'indicateur suivant cas particulier
? = résultat indéterminé / 0 = mis à zéro / 1 = mis à un.

INSTRUCTIONS	S	Z		H		P/O	N	C
ADC A,op	M	M		M		M	0	M
ADC HL,rr	M	M		?		M	0	M
ADD A,op	M	M		M		M	0	M
ADD HL,rr / ADD IX,rr / ADD IY,rr				?			0	M
AND	M	M		1		M	0	0
BIT	?	M		1		?	0	
CALL								
CCF				?			0	M

INSTRUCTIONS	S	Z		H		P/O	N	C
CP	M	M		M		M	1	M
CPD / CPDR / CPI / CPIR	M	M		M		P	1	
CPL							1	
DAA	M	M		M		M		M
DEC op	M	M		M		M	1	
DEC rr / DEC IX / DEC IY								
DI / DJNZ / EI / EX/EXX / HALT / IM								
IN r,(C)	M	M		M		M	0	
IN A,(d8)								
INC r	M	M		M		M	0	
INC rr / INC IX / INC IY								
INC (HL) / INC (IX+d8)/INC (IY+d8)	M	M		M		M	0	
IND / INI	?	P		?		?	1	
INDR / INIR	?	1		?		?	1	
JP / JR / LD								
LD A,R	M	M		0		P	0	
LDD / LDI				0		P	0	
LDDR / LDIR				0		0	0	
NEG	M	M		M		M	1	M

INSTRUCTIONS	S	Z		H		P/0	N	C
NOP								
OR	M	M		0		M	0	0
OTDR / OTIR	?	1		?		?	1	
OUT								
OUTD / OUTI	?	P		?		?	1	
POP/PUSH/RES/RET/RETI/RETN								
RL	M	M		0		M	0	M
RLA / RLCA / RRA / RRCA				0			0	M
RLC / RLD / RR / RRC / RRD	M	M		0		M	0	M
RST								
SBC A,op	M	M		M		M	1	M
SBC HL,rr	M	M		?		M	1	M
SET								
SLA / SRA / SRL	M	M		0		M	0	M
SUB	M	M		M		M	1	M
XOR	M	M		0		M	0	0

Nota :

Pour les cas particuliers notés P, se reporter aux ouvrages spécialisés déjà cités précédemment.

Ce tableau doit vous permettre, d'un seul coup d'œil, de repérer les indicateurs affectés par l'action de n'importe quelle instruction Z80.

CHAPITRE 2

TRAITEMENT DES NOMBRES

PRÉLIMINAIRE

Nous allons essayer de montrer, dans ce chapitre, la place non négligeable que tient le traitement des nombres en programmation assembleur, et bien évidemment dans tout autre langage.
Il nous arrive fréquemment, en Basic, de procéder à l'affichage d'un nombre, qu'il soit entier ou décimal, positif ou négatif.

De plus, l'affichage d'un nombre pourra se faire suivant une base quelconque, au gré de l'utilisateur ; en général, l'informaticien utilise les systèmes de numération en base 2, 10, et 16 ; voilà pourquoi nous nous limiterons aux conversion et affichage d'un nombre dans ces 3 bases ; ce qui importe avant tout, à mon sens, c'est la compréhension parfaite du mécanisme, parfois complexe, permettant de réaliser soi-même d'autres types de conversion et affichage de nombres.

En assembleur, ce problème se fait ressentir surtout par le fait que le programmeur doit ériger lui-même ses propres routines de conversion et affichage, ce qui, comme vous allez le constater, nécessite une approche rigoureuse et parfaite du processus.
La programmation de telles routines est souvent complexe, surtout en ce qui concerne le passage du binaire en une base différente d'une puissance de 2, telle la base 10 par exemple.
De plus, le traitement des nombres décimaux (ou flottants), accroît les difficultés de transformation dans ce type de numération.

PREMIERS SYSTÈMES DE NUMÉRATION SIMPLE À TRAITER EN ASSEMBLEUR

base 2 → nécessite le traitement des bits par groupes de 1
base 4 → nécessite le traitement des bits par groupes de 2
base 8 → nécessite le traitement des bits par groupes de 3
base 16 → nécessite le traitement des bits par groupes de 4

Programmation des routines

Il nous a semblé intéressant de présenter des routines utilisant les vecteurs mathématiques mis à la disposition de l'utilisateur. Toutefois, dans l'utilisation de ces vecteurs traitant des nombres flottants, il est souvent plus judicieux, et en tout cas moins coûteux en temps, de faire de la programmation en pur assembleur.

Aussi les exemples présentés avec utilisation de vecteurs mathématiques sont-ils plus d'ordre pédagogique que pratique ; de plus, l'absence de ces vecteurs permet une portabilité plus aisée entre Amstrad, du fait des différences d'implantation d'un Amstrad CPC sur l'autre.

Il est toutefois important de noter la disparition des vecteurs mathématiques traitant l'entier, avec l'apparition des CPC 664 et 6128 ; c'est pour cette raison que nous avons opté pour l'utilisation des seuls vecteurs mathématiques traitant les nombres flottants.

Dans les pages suivantes, vous trouverez un tableau présentant les vecteurs mathématiques des CPC 464, 664 et 6128.

Routines traitées dans ce chapitre

(1) conversion et affichage de nombres entiers positifs sur deux octets, en décimal, avec utilisation de vecteurs en RAM.

(2) conversion et affichage de nombres entiers signés sur deux octets, en décimal, avec utilisation de vecteurs en RAM.

(3) idem (1), mais sans utilisation de vecteurs mathématiques.

(4) idem (2), mais sans utilisation de vecteurs mathématiques.

(5) conversion et affichage de nombres flottants signés, en décimal, avec utilisation de vecteurs mathématiques.

(6) récupération de la partie entière d'un flottant, en décimal, dans un buffer, sans utilisation de vecteurs en RAM.

(7) idem (6), mais pour la partie décimale d'un nombre flottant.

(8) conversion et affichage d'un nombre entier sur deux octets, en hexa, sans utilisation des vecteurs mathématiques.

(9) conversion et affichage d'un nombre entier sur deux octets, en binaire, sans utilisation des vecteurs en RAM.

VECTEURS MATHÉMATIQUES DES CPC 464, CPC 664 ET CPC 6128

ADRESSES			F/E	FONCTIONS PRESENTEES	REGISTRES UTILISES	OBSERVATIONS EVENTUELLES
464	664	6128				
BD6A	BD88	BD8E	F	COMPARE 2 NB	(HL)>(DE) A=1 (HL)<(DE) A=255 (HL)=(DE) A=0	REGISTRES IX ET IY DETRUITS
BD6D	BD8E	BD91	F	NEGATION FLOT	(HL) → −(HL)	
DB70	BD91	BD94	F	SIGNE D'UN FLOT	(HL) >0 A=1 (HL) <0 A=255 (HL) =0 A=0	
BD73	BD94	BD97	F	DEGRE OU RADIAN	A=0 SI RADIAN A<>0 SI DEGRE	
BD76	BD97	BD9A	F	RECUP PI EN FLOT	PI → (HL)	
BD79	BD9A	BD9D	F	RACINE CARREE	RAC(HL) → (HL)	TRAIT POSITIF
BD7C	BD9D	BDA0	F	PUISSANCE	(HL)^(DE) → (HL)	
BD7F	BDA0	BDA3	F	LOG NEPERIEN	LOG(HL) → (HL)	TRAIT POSITIF
BD82	BDA3	BDA6	F	LOG DECIMAL	LOG10(HL) → (HL)	TRAIT POSITIF
BD85	BDA6	BDA9	F	EXPONENTIEL	EXP(HL) → (HL)	
BD88	BDA9	BDAC	F	SINUS	SIN(HL) → (HL)	
BD8B	BDAC	BDAF	F	COSINUS	COS(HL) → (HL)	
BD8E	BDAF	BDB2	F	TANGEANTE	TG(HL) → (HL)	
BD91	BDB2	BDB5	F	ARCTANGEANTE	ARCTG(HL) → (HL)	
BDAC			E	ADDITION	HL + DE → HL	OVERFLOW SI > +32767 OU < −32768
BDAF			E	SOUSTRACTION	HL − DE → HL	IDEM CI-DESSUS

Vecteurs mathématiques des CPC 464, CPC 664 et CPC 6128 (Suite)

REMARQUES :

flottant : 5 octets / entier : 2 octets
F : trait. flottant / E: trait. entier
(rr) : zone pointée par double reg. rr
rr : double reg. contenant un nombre entier
→ : indique le sens du trait. des reg.

ADRESSES			F/E	FONCTIONS PRESENTEES	REGISTRES UTILISES	OBSERVATIONS EVENTUELLES
464	664	6128				
BD3D	BD5E	BD61	F	COPIE UN FLOT DANS UNE ZONE	(DE → (HL)	
BD40	BD61	BD64	F	ENTIER DANS FLOT	HL → (DE)	ENTIER <0 DONNE FLOT >0
DB43	BD64	BD67	F	ENTIER DANS FLOT	(HL) → (HL)	
BD46	BD67	BD6A	F	FLOT DANS ENTIER	(HL) → HL	ARRONDI AU PLUS PROCHE TOUJOURS >0
BD49	BD6A	BD6D	F	FLOT DANS ENTIER	(HL) → (HL)	IDEM CI-DESSUS
BD4C	BD6D	BD70	F	FONCT BASIC FIX	(HL) → (HL)	IDEM CI-DESSUS
BD4F	BD70	BD73	F	FONCT BASIC INT	(HL) → (HL)	ARRONDI PAR DEFAUT TOUJOURS >0
BD58	BD79	BD7C	F	ADDITION	(HL)+(DE) → (HL)	RESPECTE SGN
BD5B	BD7F	BD82	F	SOUSTRACTION	(HL)−(DE) → (HL)	RESPECTE SGN
BD5E			E	SOUSTRACTION	(DE)−(HL) → (HL)	RESPECTE SGN
BD61	BD82	BD85	F	MULTIPLICATION	(HL)×(DE) → (HL)	RESPECTE SGN
BD64	BD85	BD88	F	DIVISION	(HL) : (DE) → (HL)	RESPECTE SGN

Vecteurs mathématiques des CPC 464 CPC 664 et CPC 6128 (suite)

ADRESSES 464	664	6128	F/E	FONCTIONS PRESENTEES	REGISTRES UTILISES	OBSERVATIONS EVENTUELLES
BDB2			E	SOUSTRACTION	DE − HL → DE	OVERFLOW SI > +32767 OU < −32768
BDB5			E	MULTIPLICATION	HL × DE → DE	IDEM CI-DESSUS
DBB8			E	DIVISION	HL : DE → HL	IDEM CI-DESSUS
BDBB			E	MODULO	HL MOD DE→HL	
BDC4			E	COMPARE 2 NB	HL>DE A=1 HL<DE A=255 HL=DE A=0	
BDC7			E	NEGATION	65536−HL → HL	
BDCA			E	SIGNE D'UN ENT	HL>0 A=1 HL<0 A=255 HL=0 A=0	

Autres routines existantes

BD97	BDB8	BDBB	F	INIT RND		
BD9A	BDBB	BDBE	F	CLASSE ALEATOIRE		
BD9D	BD7C	BD7F	F	DERNIERE VAL RND		
BD55	BD76	BD79	F	NB × 10^A		
BD67			F	NB × 2^A		
BD43	BD64	BD67	F	ENTIER EN FLOT	(HL) → (HL)	
BD94	BDB5	BDB8	F	4 OCTETS EN FLOT		

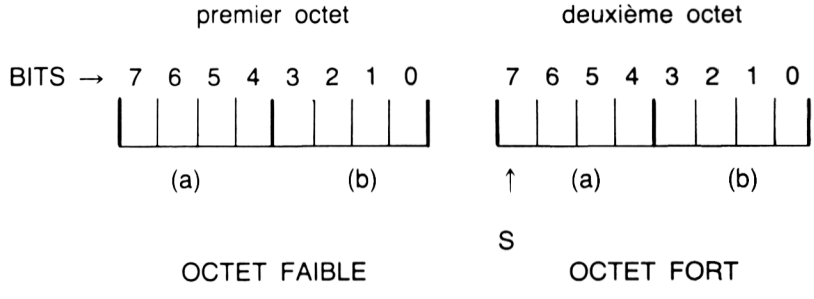
ROUTINES DE TRAITEMENT DES NOMBRES ENTIERS 16 BITS

Rappels

L'Amstrad CPC permet de représenter des nombres entiers sur un ou deux octets, soit sur 8 ou 16 bits.
Ces valeurs numériques peuvent être signées ou non, c'est-à-dire positives ou négatives.
Les valeurs numériques entières non signées sur un octet peuvent prendre les valeurs de 0 à 255.
Les valeurs numériques entières signées peuvent prendre quant à elles les valeurs de − 128 à + 127.
Dans le cas d'une valeur signée sur un octet, le bit 7 indique le signe de l'entier ; si celui-ci est à zéro l'entier est positif, sinon la valeur 1 du bit 7 indique que nous avons affaire à un entier négatif.

En ce qui concerne les valeurs numériques non signées sur deux octets, les valeurs peuvent aller de 0 à 65535.
Pour les valeurs numériques signées sur deux octets, nous pouvons obtenir des valeurs entre − 32768 et + 32767.
Dans le cas de valeur numérique signée sur deux octets, le bit de signe se situe dans le bit 7 de l'octet fort (voir ci-après).

Représentation d'un entier 16 bits en mémoire



Les 4 premiers bits d'un octet constituent le QUARTET GAUCHE : (a)
Les 4 derniers bits d'un octet constituent le QUARTET DROIT : (b)

"S" représente le SIGNE de l'entier 16 bits (0 ou 1).

Les entiers négatifs

Ceux qui ne sont pas familiarisés avec le système binaire me diront, à juste titre, qu'il ne paraît pas aisé de déterminer la valeur absolue d'un nombre entier négatif sur 8 ou 16 bits.

Le principe en est très simple, et s'appelle le "complément à 2". Mais avant de déterminer la valeur absolue d'un entier de valeur négative, il nous faut d'abord connaître la représentation d'un entier négatif en mémoire.

Essayer tout d'abord de faire PRINT HEX$(– 1), et vous obtiendrez #FFFF comme résultat, soit 1111111111111111 en binaire.
Si vous faites cette fois un PRINT HEX$ (–32768), vous obtiendrez alors #8000 comme résultat, soit 1000000000000000 en binaire.
Ces deux valeurs représentent les valeurs limites pouvant être obtenues pour un entier négatif sur 16 bits.

Vous remarquerez au passage que le premier bit est à 1 ; c'est en effet ce fameux bit de signe qui nous indique que l'entier est de valeur négative.
On remarquera également que l'affichage d'un entier, en hexa ou en binaire, place l'octet fort avant l'octet faible ; c'est pour cette raison que le bit de signe est le premier des 16 bits.

Cette représentation des nombres entiers est faite en complément à deux, ce qui nécessite l'application de ce principe pour passer d'un entier négatif à sa valeur absolue.

Voyons comment à partir de la valeur binaire nous pouvons obtenir la valeur décimale d'un entier négatif ; reprenons pour cela la valeur binaire du nombre –1 :

binaire :	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
poids →	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Le premier bit est à 1, donc l'entier est négatif ; nous devons alors effectuer la multiplication de –1 par 2 puissance 15. Tous les autres bits seront multipliés par 2 puissance "poids".

Nous obtenons alors : – 32768 + 32767 = –1

Cette méthode est toutefois lourde à utiliser par un micro, c'est pourquoi les micros utilisent le complément à 2 qui est beaucoup plus simple à traiter par la logique binaire des ordinateurs.

Voyons maintenant comment s'effectue le complément à deux :

première étape : transformation des bits 0 en 1

transformation des bits 1 en 0

deuxième étape : ajouter 1 à la valeur binaire

obtenue après transformation

Appliquons ce principe sur notre exemple de départ, soit –1 :

valeur binaire de l'entier négatif → 1111111111111111
valeur binaire après transformation → 0000000000000000
valeur binaire après ajout de 1 → 0000000000000001

Appliquons ce principe sur un autre exemple, soit sur −32767 :

valeur binaire de l'entier négatif → 1000000000000001
valeur binaire après transformation → 0111111111111110
valeur binaire après ajout de 1 → 0111111111111111

N'oubliez pas que la plus grande valeur positive pouvant être représentée sur 16 bits est +32767.

Tableau de quelques valeurs entières négatives en complément à 2 :

COMP A 2	DEC	COMP A 2	DEC	COMP A 2	DEC	COMP A 2	DEC
11111111	−1	11110011	−13	11100111	−25	11011011	−37
11111110	−2	11110010	−14	11100110	−26	11011010	−38
11111101	−3	11110001	−15	11100101	−27	11011001	−39
11111100	−4	11110000	−16	11100100	−28	11011000	−40
11111011	−5	11101111	−17	11100011	−29	11010111	−41
11111010	−6	11101110	−18	11100010	−30	11010110	−42
11111001	−7	11101101	−19	11100001	−31	11010101	−43
11111000	−8	11101100	−20	11100000	−32	11010100	−44
11110111	−9	11101011	−21	11011111	−33	11010011	−45
11110110	−10	11101010	−22	11011110	−34	11010010	−46
11110101	−11	11101001	−23	11011101	−35	11010001	−47
11110100	−12	11101000	−24	11011100	−36	11010000	−48

Affichage de nombres entiers 16 bits, en binaire

C'est l'affichage le plus simple car il ne nécessite pas de conversion, puisque toute information est représentée en binaire dans un ordinateur.

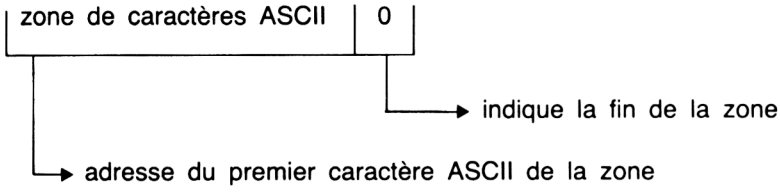
Cependant, en raison de l'organisation interne de représentation d'une valeur 16 bits, il faudra procéder en 2 étapes successives, et similaires :

Traitement de l'octet faible, puis de l'octet fort de l'entier.

Pour chaque octet traité, nous devons récupérer successivement chaque bit, du bit 0 au bit 7.
Ceci se fera simplement par des décalages à droite dans le carry. Après chaque décalage à droite, il suffira de tester le bit de report pour savoir s'il est à 0 ou à 1 ; puis, suivant le cas, nous remplirons un buffer, en partant de la droite, avec une valeur de 0 ou 1 (#30 ou #31).

Pour afficher une zone de caractères ASCII, nous disposons d'une routine en ROM, d'utilisation aisée (voir tome 14 des édit. M-A).

Rappel sur l'utilisation de cette routine :



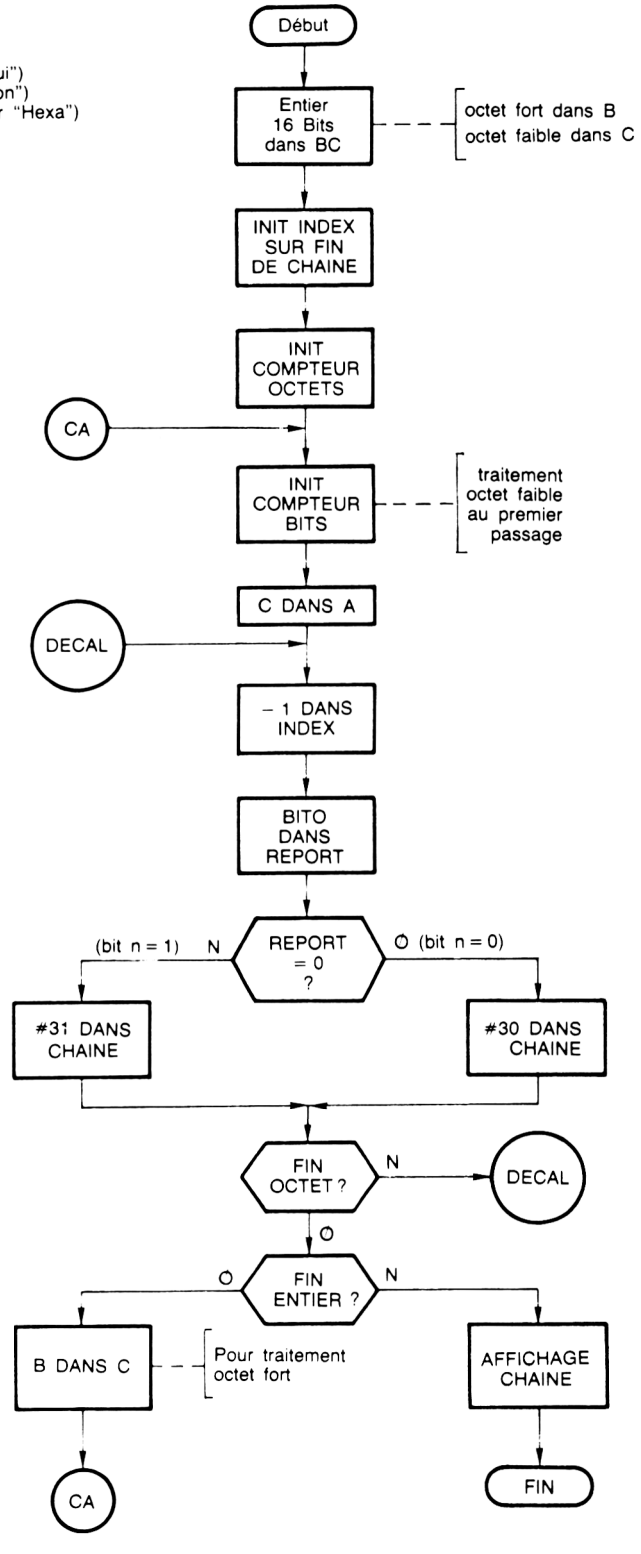
(1) se connecter en ROM HAUTE : CALL #B900
(2) fournir l'adresse du premier caractère dans le registre HL
(3) exécution de la routine d'affichage :

CALL #C337 pour le CPC 464
CALL #C380 pour le CPC 664
CALL #C37D pour le CPC 6128

Cette routine d'affichage sera utilisée tout au long de ce livre.

Routine d'affichage d'un nombre entier 16 bits en binaire

Légende :
0 → chiffre zéro
Ø → lettre (pour "oui")
N → lettre (pour "non")
→ caractère (pour "Hexa")



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                 10 ;=* ROUTINE D'AFFICH BINAIRE D'ENTIER 16 BITS *=
7000             20         ent  #7000
7000             30         org  #7000
B900             40 romhau: equ  #b900
C37D             50 affich: equ  #c37d
                 60 ;
7000  CD00B9     70         call romhau
                 80 ;--* ENTIER A TRAITER DANS BC *--
7003  ED4B3470   90         ld   bc,(entier)
                100 ;--* INIT PTR CHAINE SUR FIN TEXTE *--
7007  DD214670  110         ld   ix,fintxt
                120 ;--* INIT COMPTEUR OCTETS *--
700B  1E01      130         ld   e,1
                140 ;--* INIT COMPTEUR BITS *--
700D  1607      150 ca:     ld   d,7
                160 ;--* TRAIT OCTET FAIBLE *--
700F  79        170         ld   a,c
7010  DD2B      180 decal:  dec  ix
                190 ;--* ACQUIS BIT 0 DANS REPORT *--
7012  CB3F      200         srl  a
7014  3806      210         jr   c,bun
                220 ;--* TRAIT BIT 0 *--
7016  DD360030  230         ld   (ix+0),#30
701A  1804      240         jr   suiv
                250 ;--* TRAIT BIT 1 *--
701C  DD360031  260 bun:    ld   (ix+0),#31
                270 ;--* TEST SI FIN OCTET *--
7020  15        280 suiv:   dec  d
7021  CB7A      290         bit  7,d
                300 ;--* SI NON TRAIT BIT SUIVANT *--
7023  28EB      310         jr   z,decal
                320 ;--* TEST SI FIN ENTIER *--
7025  1D        330         dec  e
7026  CB7B      340         bit  7,e
7028  2003      350         jr   nz,fin
                360 ;--* SI NON TRAIT OCTET FORT *--
702A  48        370         ld   c,b
702B  18E0      380         jr   ca
702D  213670    390 fin:    ld   hl,chaine
7030  CD7DC3    400         call affich
7033  C9        410         ret
                420 ;
                430 ;--* ZONES DE DONNEES UTILISEES *--
7034  3BF9      440 entier: defw #f93b
7036            450 chaine: defs 16
7046            460 fintxt: defs 1
```

```
Pass 2 errors: 00

affich C37D  bun    701C  ca     700D
chaine 7036  decal  7010  entier 7034
fin    702D  fintxt 7046  romhau B900
suiv   7020

Table used:   130  from   900
Executes: 28672
```

Affichage de nombres entiers 16 bits, en hexadécimal

Cet affichage nécessite une conversion, puisque nous devons transformer chaque quartet (4 bits) en une valeur hexa (0 à F).

Comme pour l'affichage en binaire, nous traiterons l'octet faible en premier, puis l'octet fort.
Pour chaque octet, nous traiterons successivement le quartet de poids faible (quartet droit), puis le quartet de poids fort (quartet gauche).

Pour traiter le quartet droit, nous mettrons à zéro le quartet gauche, puis nous examinerons la valeur de l'octet ainsi obtenu.

En ce qui concerne le traitement du quartet gauche, il faudra procéder en deux étapes :

(a) mise du quartet gauche dans le quartet droit ;
(b) mise à zéro du quartet droit.

Pour chaque octet ainsi obtenu, nous devrons examiner sa valeur.

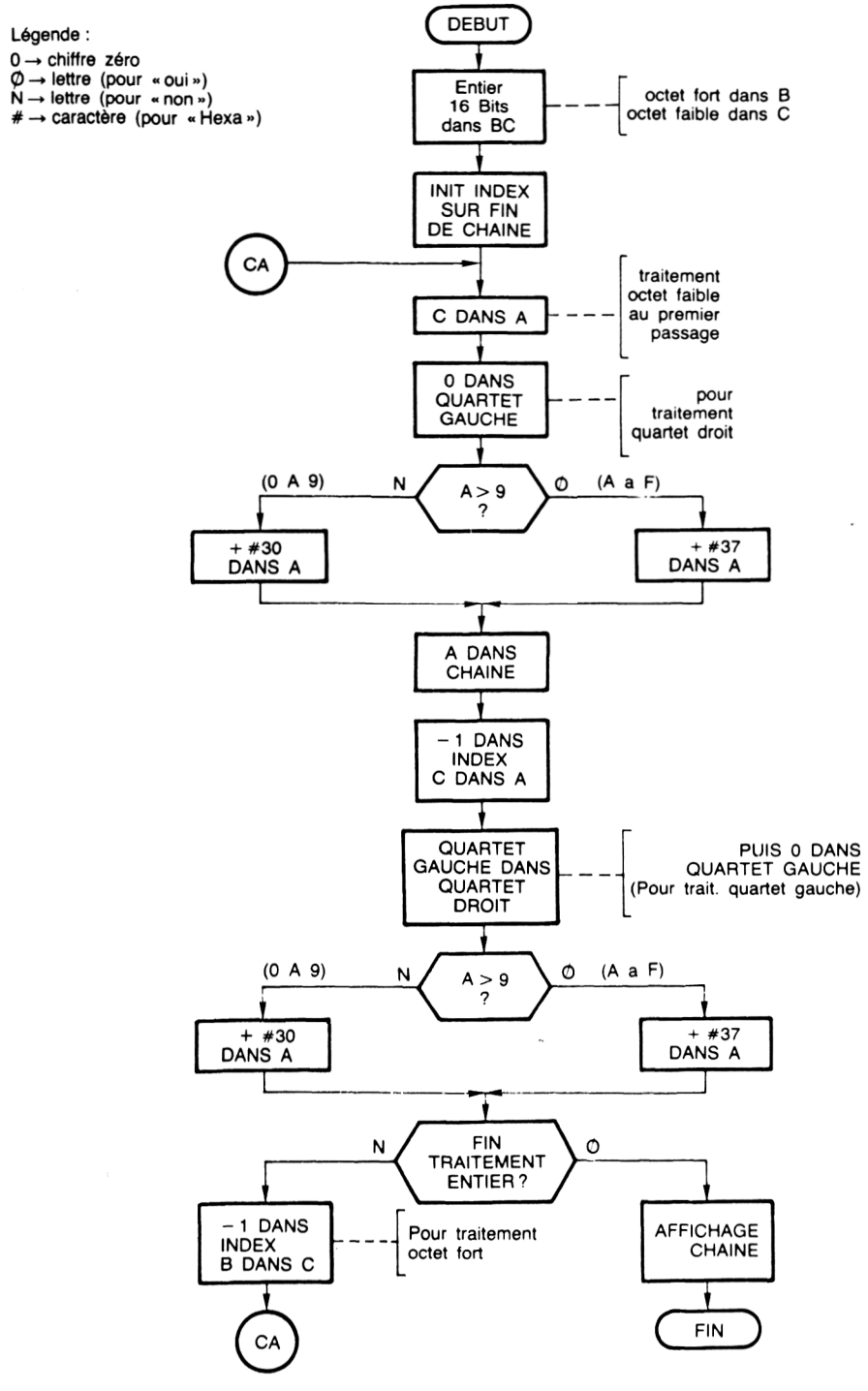
Deux cas vont se présenter :

(a) l'octet indique que le quartet traité a une valeur comprise entre 0 et 9 ; dans ce cas, nous devrons ajouter #30 pour obtenir le chiffre ASCII voulu.
(b) l'octet indique que le quartet traité a une valeur comprise entre 10 et 15 ; dans ce cas, nous devrons ajouter #37 pour obtenir le caractère ASCII voulu, soit A à F.

De même que dans l'affichage en binaire, nous rangerons successivement chaque caractère ASCII ainsi obtenu dans un buffer, en partant de la droite de celui-ci. Pour le traitement d'un entier 16 bits, nous obtiendrons quatre caractères ASCII dans le buffer.

Ci-après, vous trouverez un organigramme de programmation, puis la routine correspondante en assembleur Z80.

Routine de conversion et d'affichage d'un nombre entier 16 bits en Hexa



```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                 10 ;=* ROUTINE D'AFFICHAGE HEXA D'UN ENTIER 16 BITS *=
7000             20         ent  #7000
7000             30         org  #7000
B900             40 romhau: equ  #b900
C37D             50 affich: equ  #c37d
                 60 ;
7000  CD00B9     70         call romhau
                 80 ;--* RECUP ENTIER A TRAITER *--
7003  ED4B4470   90         ld   bc,(entier)
                100 ;--* INIT PTR CHAINE SUR FIN CHAINE *--
7007  DD214A70  110         ld   ix,fintxt
700B  1E01      120         ld   e,1
                130 ;--* TRAITEMENT OCTET FAIBLE *--
700D  79        140 ca:     ld   a,c
                150 ;--* TRAITEMENT QUARTET FAIBLE *--
700E  E60F      160         and  #0f
7010  FE0A      170         cp   10
                180 ;--* ALLER A ZAN1 POUR TRAIT 0 A 9 *--
7012  3802      190         jr   c,zan1
                200 ;--* TRAIT CHIFFRE 'A' A 'F' *--
7014  CE07      210         adc  a,7
7016  A7        220 zan1:   and  a
7017  CE30      230         adc  a,#30
                240 ;--* -1 DANS POINTEUR CHAINE *--
7019  DD2B      250         dec  ix
                260 ;--* CHIFFRE ASCII DANS CHAINE *--
701B  DD7700    270         ld   (ix+0),a
                280 ;--* TRAITEMENT QUARTET FORT *--
701E  79        290         ld   a,c
                300 ;--* QUART FORT DANS QUART FAIBLE *--
701F  CB3F      310         srl  a
7021  CB3F      320         srl  a
7023  CB3F      330         srl  a
7025  CB3F      340         srl  a
7027  FE0A      350         cp   10
                360 ;--* ALLER A ZAN2 POUR TRAIT 0 A 9 *--
7029  3802      370         jr   c,zan2
                380 ;--* TRAIT CHIFFRE 'A' A 'F' *--
702B  CE07      390         adc  a,7
702D  A7        400 zan2:   and  a
702E  CE30      410         adc  a,#30
                420 ;--* -1 DANS POINTEUR CHAINE *--
7030  DD2B      430         dec  ix
                440 ;--* CHIFFRE ASCII DANS CHAINE *--
7032  DD7700    450         ld   (ix+0),a
                460 ;--* TEST SI FIN TRAIT OCTETS *--
```

```
7035  1D        470          dec  e
7036  CB7B      480          bit  7,e
7038  2003      490          jr   nz,fin
                500 ;--* ALLER TRAITER OCTET FORT *--
703A  48        510          ld   c,b
703B  18D0      520          jr   ca
                530 ;--* AFFICHAGE HEXA DE L'ENTIER *--
703D  214670    540 fin:     ld   hl,chaine
7040  CD7DC3    550          call affich
7043  C9        560          ret
                570 ;
                580 ;--* ZONES DE DONNEES UTILISEES *--
7044  E3F0      590 entier: defw #f0e3
7046            600 chaine: defs 4
704A            610 fintxt: defs 1

Pass 2 errors: 00

affich C37D  ca     700D  chaine 7046
entier 7044  fin    703D  fintxt 704A
romhau B900  zan1   7016  zan2   702D


Table used:   119  from   900
Executes: 28672
```

Affichage de nombres entiers 16 bits, signés ou non, en décimal

Il existe plusieurs méthodes de conversion pour obtenir une chaîne de chiffres décimaux à partir d'un nombre binaire.

Nous n'en retiendrons que 2 qui sont :

— méthode de la division par 10,
— méthode de comptage des dizaines.

MÉTHODE DE LA DIVISION PAR 10

Cette méthode consiste à isoler chaque dizaine, en effectuant des divisions successives par 10 ; pour un nombre allant jusqu'aux dizaines de milliers, nous obtiendrons successivement :

— les unités,
— les dizaines,
— les centaines,
— les milliers,
— les dizaines de mille.

Pour chaque chiffre ainsi obtenu, il suffira de les ranger dans un buffer, en partant de la droite de celui-ci.

Prenons un exemple : soit le nombre entier non signé 12345.

Après divisions successives par 10, nous obtiendrons, dans l'ordre d'exécution, les chiffres : 5 / 4 / 3 / 2 / 1.

Plutôt qu'un long discours, nous allons disséquer le processus à partir de l'exemple ci-dessus.
Pour obtenir chaque chiffre, nous exécuterons 4 étapes :

(1) division d'un entier par 10 (voir étapes),
(2) récup. partie entière du résultat précédent,
(3) multiplication résultat précédent par 10,
(4) soustract. entier par résultat précédent.

Obtention des unités : (entier au départ = 12345)

première étape → 12345 / 10 = 1234.5
deuxième étape → int (1234.5) = 1234 (à conserver)
troisième étape → 1234 × 10 = 12340
quatrième étape → 12345 − 12340 = 5 (unité)

Obtention des dizaines : (nouvel entier = 1234)

première étape → 1234 / 10 = 123.4
deuxième étape → int (123.4) = 123 (à conserver)
troisième étape → 123 × 10 = 1230
quatrième étape → 1234 − 1230 = 4 (dizaine)

Obtention des centaines : (nouvel entier = 123)

première étape → 123 / 10 = 12.3
deuxième étape → int (12.3) = 12 (à conserver)
troisième étape → 12 × 10 = 120
quatrième étape → 123 − 120 = 3 (centaine)

Obtention des milliers : (nouvel entier = 12)

première étape → 12 / 10 = 1.2
deuxième étape → int (1.2) = 1 (à conserver)
troisième étape → 1 × 10 = 10
quatrième étape → 12 − 10 = 2 (millier)

Obtention des dizaines de mille : (nouvel entier = 1)

première étape → 1 / 10 = 0.1
deuxième étape → int (0.1) = 0 (à conserver)
troisième étape → 0 × 10 = 0
quatrième étape → 1 − 0 = 1 (dizaine de mille).

Nous pouvons remarquer que le résultat de la deuxième étape est systématiquement conservé ; cet entier est reporté en début du traitement de l'obtention du chiffre suivant.

Toutefois une question se pose : comment allons nous savoir que le traitement est fini ?
La réponse en est fort simple ; en effet, après chaque passage, nous allons tester si le résultat de la deuxième étape est égal à 0, auquel cas nous venons d'obtenir le dernier chiffre de l'entier.

Pour effectuer ce processus, nous utiliserons les vecteurs en RAM.

En ce qui concerne le traitement d'un entier signé, il suffira de déterminer au départ le signe de l'entier 16 bits, en testant le bit 7 de l'octet fort de l'entier ; le signe ainsi déterminé sera mis en début du buffer.

Deux routines assembleur sont présentées ; la première permet de ne traiter que les entiers 16 bits non signés ; la deuxième permet de traiter les entiers 16 bits signés.

Les routines présentées tournent sur CPC 6128 ; pour les faire tourner sur CPC 464, il vous suffit d'effectuer, outre les divers changements de vecteurs mathématiques, une petite modification. Celle-ci est due à une modification du système d'exploitation, en ce qui concerne le traitement du flottant, entre le CPC 464 et les CPC 664 / 6128. En effet, une valeur flottante à 0 sur les CPC 664 et CPC 6128 donne un exposant nul, alors que pour le CPC 464 cet exposant est bien à #80 ; il convient donc de corriger celui-ci pour les CPC 664 et 6128, en y mettant la valeur #80.

Adaptation à faire pour CPC 464 (outre les vecteurs RAM)

(a) pour la routine de traitement des entiers non signés :

Supprimer les instructions 890 à 930, ainsi que les instructions 1300 à 1320.

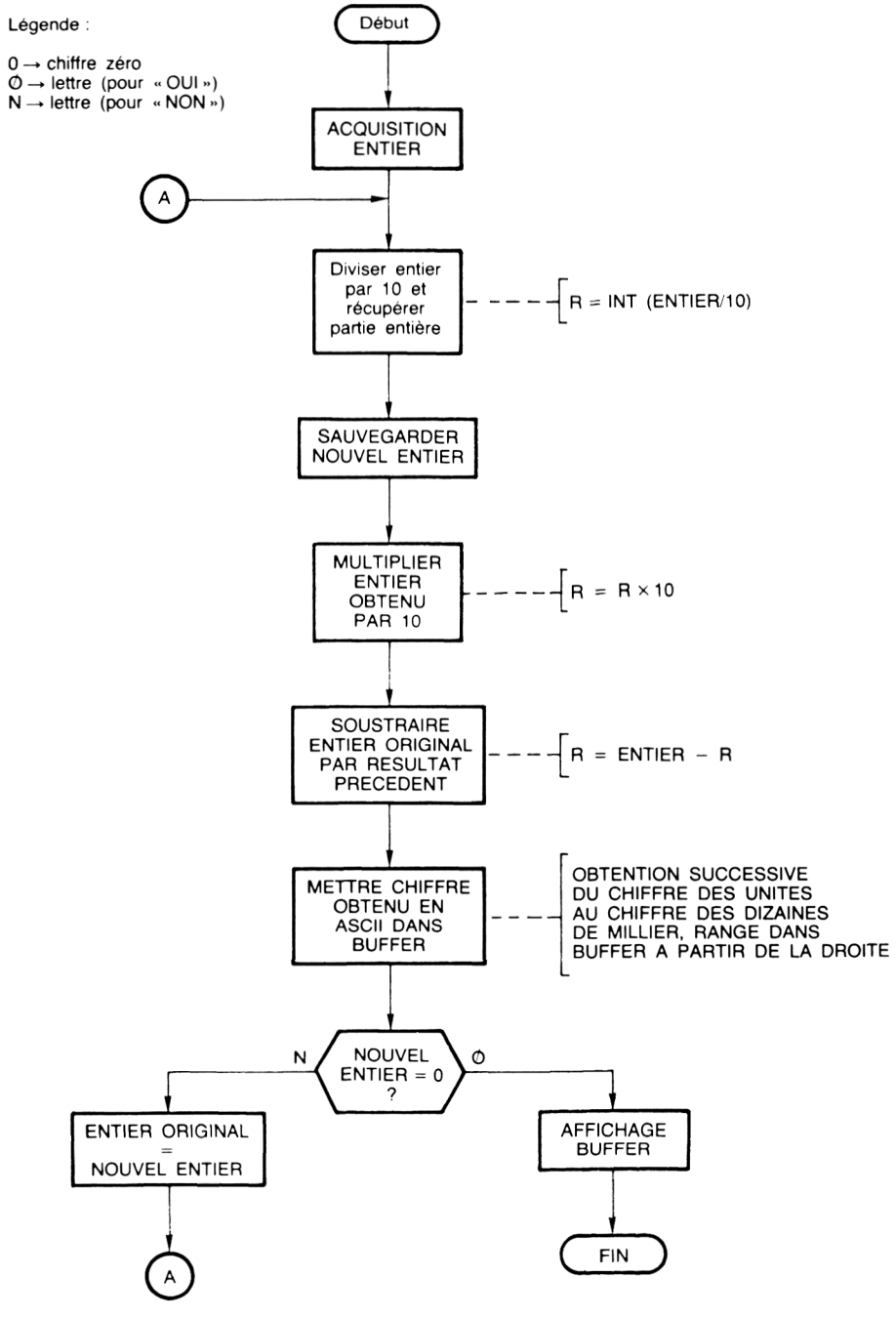
(b) pour la routine de traitement des entiers signés :

Supprimer les instructions 1040 à 1080, ainsi que les instructions 1570 à 1590.

Cette méthode utilise 237 octets d'assembleur, et 30 octets de données pour le traitement de l'entier non signé.
En ce qui concerne le traitement de l'entier signé, la routine utilise 271 octets pour les instructions assembleur, et 32 pour les données.

Ci-après, vous trouverez les 2 routines assembleur, accompagnées de leurs organigrammes de programmation.

Routine de conversion et affichage d'un nombre entier 16 bits non signé, en décimal



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                 10 ;=* ROUT AFF VALEURS NUM POSITIVES ENTIERES *=
                 20 ;=* SUR 16 BITS / POUR CPC 6128 *=
7000             30         ent  #7000
7000             40         org  #7000
                 50 ;--* CONNECTION EN ROM HAUTE *--
B900             60 romhau: equ  #b900
                 70 ;--* TRANS. ENTIER EN FLOTTANT *--
BD64             80 entflo: equ  #bd64
                 90 ;--* RECUP PARTIE ENTIERE D'UN FLOTTANT *--
BD73            100 intflo: equ  #bd73
                110 ;--* OPERATIONS EN FLOTTANT *--
BD88            120 divise: equ  #bd88
BD85            130 multip: equ  #bd85
BD82            140 soustr: equ  #bd82
BD7C            150 additi: equ  #bd7c
                160 ;--* TRANS. FLOTTANT EN ENTIER *--
BD6A            170 floent: equ  #bd6a
                180 ;--* ROUTINE D'AFFICHAGE D'UN TEXTE ASCII *--
C37D            190 affich: equ  #c37d
7000  CD00B9    200         call romhau
                210 ;--* INITIALISATION BUFFER A BLANC *--
7003  3E05      220         ld   a,5
7005  DD21EF70  230         ld   ix,buffer
7009  DD360020  240 blbuf:  ld   (ix+0),#20
700D  DD23      250         inc  ix
700F  3D        260         dec  a
7010  20F7      270         jr   nz,blbuf
                280 ;--* INITIALISATION POINTEUR BUFFER *--
7012  01F470    290         ld   bc,finbuf
7015  ED430A71  300         ld   (pb),bc
                310 ;--* INIT. FIN BUFFER A ZERO POUR ROUTINE AFF. *--
7019  3E00      320         ld   a,0
701B  32F470    330         ld   (finbuf),a
                340 ;--* INIT. DES ZONES ACCUM1 ET ACCUM2 A ZERO *--
701E  3E0A      350 chiffr: ld   a,10
7020  DD21FA70  360         ld   ix,accum1
7024  DD360000  370 mazacc: ld   (ix+0),0
7028  DD23      380         inc  ix
702A  3D        390         dec  a
702B  20F7      400         jr   nz,mazacc
                410 ;--* RECUP. ENTIER EN FLOTTANT *--
702D  2AED70    420         ld   hl,(entier)
7030  11FA70    430         ld   de,accum1
7033  CD64BD    440         call entflo
                450 ;--* SAUVE FLOTTANT EN STO *--
```

```
7036  21FA70     460          ld   hl,accum1
7039  11F570     470          ld   de,sto
703C  010500     480          ld   bc,5
703F  EDB0       490          ldir
                 500 ;--* 10 DANS ACCUM2 *--
7041  2A0671     510          ld   hl,(dix)
7044  11FF70     520          ld   de,accum2
7047  CD64BD     530          call entflo
                 540 ;--* ACCUM1 = ACCUM1 / 10 *--
704A  21FA70     550          ld   hl,accum1
704D  11FF70     560          ld   de,accum2
7050  CD88BD     570          call divise
                 580 ;--* RECUP PARTIE ENTIERE DE ACCUM1 *--
7053  21FA70     590          ld   hl,accum1
7056  CD73BD     600          call intflo
                 610 ;--* SAUVE PARTIE ENTIERE DANS STOENT *--
7059  ED4BFA70   620          ld   bc,(accum1)
705D  ED430871   630          ld   (stoent),bc
                 640 ;--* TRANS. ENTIER STOENT EN FLOT. DANS ACCUM1 *--
7061  2A0871     650          ld   hl,(stoent)
7064  11FA70     660          ld   de,accum1
7067  CD64BD     670          call entflo
                 680 ;--* 10 DANS ACCUM2 *--
706A  2A0671     690          ld   hl,(dix)
706D  11FF70     700          ld   de,accum2
7070  CD64BD     710          call entflo
7073  1802       720          jr   suite
                 730 ;--* PONT POUR REMONTEE A CHIFFR (RELOGEABLE) *--
7075  18A7       740 ch:      jr   chiffr
                 750 ;--* ACCUM1 = ACCUM1 * 10 *--
7077  11FF70     760 suite:   ld   de,accum2
707A  21FA70     770          ld   hl,accum1
707D  CD85BD     780          call multip
                 790 ;--* ACCUM1 DANS ACCUM2 *--
7080  21FA70     800          ld   hl,accum1
7083  11FF70     810          ld   de,accum2
7086  010500     820          ld   bc,5
7089  EDB0       830          ldir
                 840 ;--* STO DANS ACCUM1 *--
708B  21F570     850          ld   hl,sto
708E  11FA70     860          ld   de,accum1
7091  010500     870          ld   bc,5
7094  EDB0       880          ldir
                 890 ;--* TEST ET BRANCH. ST EXP. ACCUM2 A ZERO *--
7096  DD21FF70   900          ld   ix,accum2
709A  DD7E04     910          ld   a,(ix+4)
709D  FE00       920          cp   0
709F  CCE870     930          call z,exp80
                 940 ;--* ACCUM1 = ACCUM1 - ACCUM2 *--
```

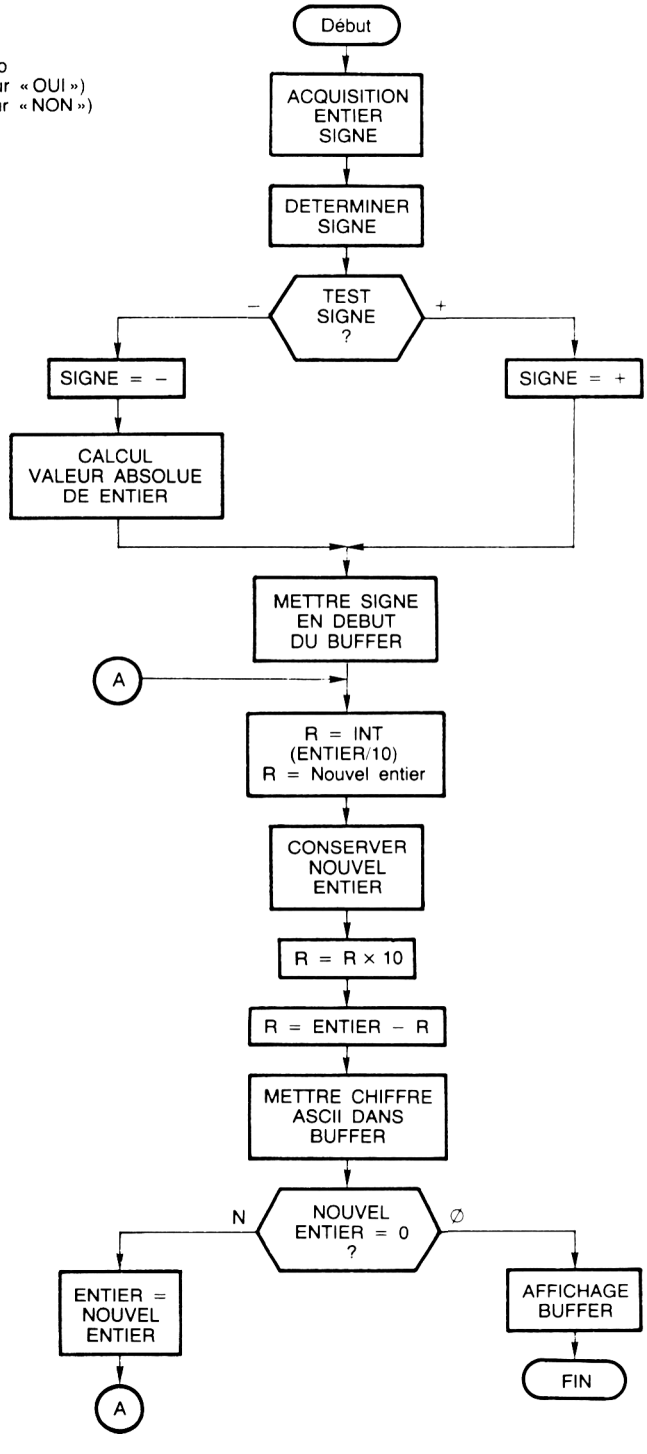
```
70A2  21FA70     950          ld   hl,accum1
70A5  11FF70     960          ld   de,accum2
70A8  CD82BD     970          call soustr
                 980 ;--* #30 DANS ACCUM2 *--
70AB  2A0471     990          ld   hl,(trente)
70AE  11FF70    1000          ld   de,accum2
70B1  CD64BD    1010          call entflo
                1020 ;--* ACCUM1 = ACCUM1 + ACCUM2 (TRANS. ASCII) *--
70B4  21FA70    1030          ld   hl,accum1
70B7  11FF70    1040          ld   de,accum2
70BA  CD7CBD    1050          call additi
                1060 ;--* RECUP ASCII DANS HL *--
70BD  21FA70    1070          ld   hl,accum1
70C0  CD73BD    1080          call intflo
70C3  2AFA70    1090          ld   hl,(accum1)
                1100 ;--* POINT BUF = POINT BUF - 1 *--
70C6  ED4B0A71  1110          ld   bc,(pb)
70CA  0B        1120          dec  bc
70CB  ED430A71  1130          ld   (pb),bc
                1140 ;--* TRANS. ASCII DANS BUFFER *--
70CF  7D        1150          ld   a,l
70D0  2A0A71    1160          ld   hl,(pb)
70D3  77        1170          ld   (hl),a
                1180 ;--* TEST ET BRANCH. SI STOENT = 0 *--
70D4  2A0871    1190          ld   hl,(stoent)
70D7  7D        1200          ld   a,l
70D8  FE00      1210          cp   0
70DA  2805      1220          jr   z,affbuf
                1230 ;--* SINON ENTIER = ASCII (HL) *--
70DC  22ED70    1240          ld   (entier),hl
70DF  1894      1250          jr   ch
                1260 ;--* AFFICHAGE BUFFER *--
70E1  21EF70    1270 affbuf: ld   hl,buffer
70E4  CD7DC3    1280          call affich
70E7  C9        1290          ret
                1300 ;--* CORRECTION EXPOSANT A #80 *--
70E8  DD360480  1310 exp80:  ld   (ix+4),#80
70EC  C9        1320          ret
                1330 ;--* ZONES UTILISEES PAR LA ROUTINE *--
70ED  3930      1340 entier: defw 12345
70EF            1350 buffer: defs 5                    ;BUFFER POUR ASCII
70F4            1360 finbuf: defs 1                    ;FIN BUFFER
70F5            1370 sto:    defs 5                    ;STOCKAGE FLOTTANT
70FA            1380 accum1: defs 5                    ;ZONE DE CALCUL
70FF            1390 accum2: defs 5                    ;ZONE DE CALCUL
7104  3000      1400 trente: defw #30
7106  0A00      1410 dix:    defw 10
7108            1420 stoent: defs 2                    ;STOCKAGE ENTIER
710A            1430 pb:     defs 2                    ;POINTEUR BUFFER
```

```
Pass 2 errors: 00

accuml 70FA  accum2 70FF  additi BD7C
affbuf 70E1  affich C37D  blbuf  7009
buffer 70EF  ch      7075 chiffr 701E
divise BD88  dix     7106 entflo BD64
entier 70ED  exp80   70E8 finbuf 70F4
floent BD6A  intflo BD73 mazacc 7024
multip BD85  pb      710A romhau B900
soustr BD82  sto     70F5 stoent 7108
suite  7077  trente 7104

Table used:   334  from   900
Executes: 28672
```

Routine de conversion et affichage d'un nombre entier 16 bits signé, en décimal

Légende :

0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → lettre (pour « NON »)



```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                  10 ;=* ROUT AFF VALEURS ENTIERES SIGNEES *=
                  20 ;=* POUR CPC 6128 *=
7000              30         ent  #7000
7000              40         org  #7000
                  50 ;--* CONNECTION EN ROM HAUTE *--
B900              60 romhau: equ  #b900
                  70 ;--* TRANS. ENTIER EN FLOTTANT *--
BD64              80 entflo: equ  #bd64
                  90 ;--* OPERATIONS EN FLOTTANT *--
BD88             100 divise: equ  #bd88
BD85             110 multip: equ  #bd85
BD82             120 soustr: equ  #bd82
BD7C             130 additi: equ  #bd7c
                 140 ;--* RECUP PARTIE ENTIERE D'UN FLOTTANT *--
BD73             150 intflo: equ  #bd73
                 160 ;--* TRANS. FLOTTANT EN ENTIER *--
BD6A             170 floent: equ  #bd6a
                 180 ;--* ROUTINE D'AFFICHAGE D'UN TEXTE ASCII *--
C37D             190 affich: equ  #c37d
7000  CD00B9     200         call romhau
                 210 ;--* INITIALISATION BUFFER A BLANC *--
7003  3E06       220         ld   a,6
7005  DD211171   230         ld   ix,buffer
7009  DD360020   240 blbuf:  ld   (ix+0),#20
700D  DD23       250         inc  ix
700F  3D         260         dec  a
7010  20F7       270         jr   nz,blbuf
                 280 ;--* INITIALISATION POINTEUR BUFFER *--
7012  011771     290         ld   bc,finbuf
7015  ED432D71   300         ld   (pb),bc
                 310 ;--* INIT. FIN BUFFER A ZERO POUR ROUTINE AFF. *--
7019  3E00       320         ld   a,0
701B  321771     330         ld   (finbuf),a
                 340 ;--* TEST SIGNE DE ENTIER *--
701E  2A0F71     350         ld   hl,(entier)
7021  E5         360         push hl
7022  CB7C       370         bit  7,h
                 380 ;--* ALLER A CHIFFR SI ENTIER POSITIF *--
7024  280A       390         jr   z,chiffr
                 400 ;--* COMPL. A 2 SI SGN ENTIER NEGATIF *--
7026  7C         410         ld   a,h
7027  2F         420         cpl
7028  67         430         ld   h,a
7029  7D         440         ld   a,l
702A  2F         450         cpl
```

```
702B  6F        460          ld   l,a
702C  23        470          inc  hl
702D  220F71    480          ld   (entier),hl
                490 ;--* TRAIT. VAL. ABS ENTIER ORIGINAL *--
7030  3E0A      500 chiffr: ld   a,10
7032  DD211D71  510          ld   ix,accum1
7036  DD360000  520 mazacc: ld   (ix+0),0
703A  DD23      530          inc  ix
703C  3D        540          dec  a
703D  20F7      550          jr   nz,mazacc
                560 ;--* RECUP. ENTIER EN FLOTTANT *--
703F  2A0F71    570          ld   hl,(entier)
7042  111D71    580          ld   de,accum1
7045  CD64BD    590          call entflo
                600 ;--* SAUVE FLOTTANT EN STO *--
7048  211D71    610          ld   hl,accum1
704B  111871    620          ld   de,sto
704E  010500    630          ld   bc,5
7051  EDB0      640          ldir
                650 ;--* 10 DANS ACCUM2 *--
7053  2A2971    660          ld   hl,(dix)
7056  112271    670          ld   de,accum2
7059  CD64BD    680          call entflo
                690 ;--* ACCUM1 = ACCUM1 / 10 *--
705C  211D71    700          ld   hl,accum1
705F  112271    710          ld   de,accum2
7062  CD88BD    720          call divise
                730 ;--* RECUP PARTIE ENTIERE DE ACCUM1 *--
7065  211D71    740          ld   hl,accum1
7068  CD73BD    750          call intflo
                760 ;--* SAUVE PARTIE ENTIERE DANS STOENT *--
706B  ED4B1D71  770          ld   bc,(accum1)
706F  ED432B71  780          ld   (stoent),bc
                790 ;--* TRANS. ENTIER STOENT EN FLOT. DANS ACCUM1 *--
7073  2A2B71    800          ld   hl,(stoent)
7076  111D71    810          ld   de,accum1
7079  CD64BD    820          call entflo
                830 ;--* 10 DANS ACCUM2 *--
707C  2A2971    840          ld   hl,(dix)
707F  112271    850          ld   de,accum2
7082  CD64BD    860          call entflo
7085  1802      870          jr   suite
                880 ;--* PONT POUR REMONTEE A CHIFFR (RELOGEABLE) *--
7087  18A7      890 ch:      jr   chiffr
                900 ;--* ACCUM1 = ACCUM1 * 10 *--
7089  112271    910 suite:  ld   de,accum2
708C  211D71    920          ld   hl,accum1
708F  CD85BD    930          call multip
                940 ;--* ACCUM1 DANS ACCUM2 *--
```

```
7092  211D71     950          ld   hl,accum1
7095  112271     960          ld   de,accum2
7098  010500     970          ld   bc,5
709B  EDB0       980          ldir
                 990 ;--* STO DANS ACCUM1 *--
709D  211871    1000          ld   hl,sto
70A0  111D71    1010          ld   de,accum1
70A3  010500    1020          ld   bc,5
70A6  EDB0      1030          ldir
                1040 ;--* TEST ET BRANCH. SI EXP. ACCUM2 A ZERO *--
70A8  DD212271  1050          ld   ix,accum2
70AC  DD7E04    1060          ld   a,(ix+4)
70AF  FE00      1070          cp   0
70B1  CC0A71    1080          call z,exp80
                1090 ;--* ACCUM1 = ACCUM1 - ACCUM2 *--
70B4  211D71    1100          ld   hl,accum1
70B7  112271    1110          ld   de,accum2
70BA  CD82BD    1120          call soustr
                1130 ;--* #30 DANS ACCUM2 *--
70BD  2A2771    1140 .        ld   hl,(trente)
70C0  112271    1150          ld   de,accum2
70C3  CD64BD    1160          call entflo
                1170 ;--* ACCUM1 = ACCUM1 + ACCUM2 (TRANS. ASCII) *--
70C6  211D71    1180          ld   hl,accum1
70C9  112271    1190          ld   de,accum2
70CC  CD7CBD    1200          call additi
                1210 ;--* RECUP ASCII DANS HL *--
70CF  211D71    1220          ld   hl,accum1
70D2  CD73BD    1230          call intflo
70D5  2A1D71    1240          ld   hl,(accum1)
                1250 ;--* POINT BUF = POINT BUF - 1 *--
70D8  ED4B2D71  1260          ld   bc,(pb)
70DC  0B        1270          dec  bc
70DD  ED432D71  1280          ld   (pb),bc
                1290 ;--* TRANS. ASCII DANS BUFFER *--
70E1  7D        1300          ld   a,l
70E2  2A2D71    1310          ld   hl,(pb)
70E5  77        1320          ld   (hl),a
                1330 ;--* TEST ET BRANCH. SI STOENT = 0 *--
70E6  2A2B71    1340          ld   hl,(stoent)
70E9  7D        1350          ld   a,l
70EA  FE00      1360          cp   0
70EC  2805      1370          jr   z,affbuf
                1380 ;--* SINON ENTIER = ASCII (HL) *--
70EE  220F71    1390          ld   (entier),hl
70F1  1894      1400          jr   ch
                1410 ;--* TEST SIGNE ENTIER ORIGINAL *--
70F3  E1        1420 affbuf: pop  hl
70F4  CB7C      1430          bit  7,h
```

```
70F6  2804      1440          jr   z,plus
                1450 ;--* SIGNE NEGATIF DANS BUFFER *--
70F8  3E2D      1460          ld   a,#2d
70FA  1802      1470          jr   sorbuf
                1480 ;--* SIGNE POSITIF DANS BUFFER *--
70FC  3E2B      1490 plus:    ld   a,#2b
70FE  2A2D71    1500 sorbuf:  ld   hl,(pb)
7101  2D        1510          dec  l
7102  77        1520          ld   (hl),a
                1530 ;--* AFFICHAGE BUFFER *--
7103  211171    1540          ld   hl,buffer
7106  CD7DC3    1550          call affich
7109  C9        1560          ret
                1570 ;--* CORRECTION EXPOSANT A #80 *--
710A  DD360480  1580 exp80:   ld   (ix+4),#80
710E  C9        1590          ret
                1600 ;--* ZONES UTILISEES PAR LA ROUTINE *--
710F  D5DD      1610 entier:  defw #ddd5
7111            1620 buffer:  defs 6                ;BUFFER (dont SGN)
7117            1630 finbuf:  defs 1                ;FIN BUFFER
7118            1640 sto:     defs 5                ;STOCKAGE FLOTTANT
711D            1650 accum1:  defs 5                ;ZONE DE CALCUL
7122            1660 accum2:  defs 5                ;ZONE DE CALCUL
7127  3000      1670 trente:  defw #30
7129  0A00      1680 dix:     defw 10
712B            1690 stoent:  defs 2                ;STOCKAGE ENTIER
712D            1700 pb:      defs 2                ;POINTEUR BUFFER


Pass 2 errors: 00

accum1 711D  accum2 7122  additi BD7C
affbuf 70F3  affich C37D  blbuf  7009
buffer 7111  ch      7087 chiffr 7030
divise BD88  dix     7129 entflo BD64
entier 710F  exp80   710A finbuf 7117
floent BD6A  intflo  BD73 mazacc 7036
multip BD85  pb      712D plus   70FC
romhau B900  sorbuf  70FE soustr BD82
sto    7118  stoent  712B suite  7089
trente 7127

Table used:   358  from   900
Executes: 28672
```

MÉTHODE DE COMPTAGE DES DIZAINES

Cette méthode consiste à compter successivement le nombre de chaque dizaine, en partant de la plus élevée, soit :

— dizaines de mille,
— milliers,
— centaines,
— dizaines,
— unités.

De même que pour la première méthode, nous allons analyser le processus sur un exemple.
Pour chaque dizaine, nous effectuerons 3 étapes successives :

(1) passage d'une constante pour comptage dizaine,
(2) soustraire la constante à un entier,
(3) tester si le résultat précédent est > 0 ou < 0.
si celui-ci est > 0, faire + 1 dans compteur et retour en (2)
si < 0, arrêt comptage, et mise compteur en ASCII en buffer.

Soit la valeur numérique entière non signée 00123.

Obtention des dizaines de mille : (entier au départ = 00123)

(1) constante = 10000,
(2) 00123 – 10000 = résultat < 0,
(3) résultat précédent < 0, arrêt comptage et mise compteur = 1 en ASCII dans buffer (dizaine de mille).

Obtention des milliers : (nouvel entier = 0123)

(1) constante = 1000,
(2) 0123 – 1000 = résultat <0,
(3) résultat précédent < 0, arrêt comptage et mise compteur = 1 en ASCII dans buffer (millier).

Obtention des centaines : (nouvel entier = 123)

(1) constante = 100,
(2) 123 — 100 = 23 (à conserver),
(3) résultat précédent > 0, compteur = 1,
(2) 23 – 100 = résultat < 0,
(3) résultat précédent < 0, arrêt comptage et mise compteur = 1 en ASCII dans buffer (centaine).

Obtention des dizaines : (nouvel entier = 23)

(1) constante = 10,
(2) 23 — 10 = 13 (à conserver),
(3) résultat précédent > 0, compteur = 1,
(2) 13 – 10 = 3 (à conserver)
(3) résultat précédent > 0, compteur = 2

(2) 3 − 10 = résultat < 0
(3) résultat précédent < 0, arrêt comptage et mise compteur = 2 en ASCII dans buffer (dizaine).

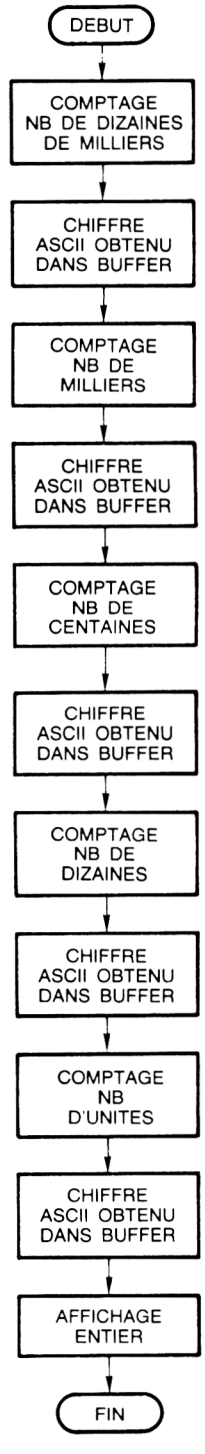
Obtention des unités : (nouvel entier = 3)

(1) constante = 1,
(2) 3 — 1 = 2 (à conserver),
(3) résultat précédent > 0, compteur = 1,
(2) 2 − 1 = 1 (à conserver)
(3) résultat précédent > 0, compteur = 2
(2) 1 − 1 = 0 (à conserver)
(3) résultat précédent < 0, compteur = 3
(2) 0 − 1 = résultat < 0
(3) résultat précédent < 0, arrêt comptage et mise compteur = 3 en ASCII dans buffer (unité).

Les 2 routines présentées ci-après, et accompagnées de leur organigramme de programmation, sont écrites pour le CPC 6128. Pour les utiliser sur votre CPC 464 ou 664, seuls les vecteurs RAM sont à changer.

On pourra remarquer que cette méthode offre des routines moins gourmandes en place mémoire ; en effet, la routine de traitement de l'entier non signé n'occupe que 67 octets en instructions, et 8 octets en données.
De même, pour la routine de traitement de l'entier signé, la place prise en instructions est de 106 octets, et seulement 9 octets pour les données.

Routine de conversion et affichage d'un entier non signés 16 bits, en décimal, sans utilisation des vecteurs mathématiques



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

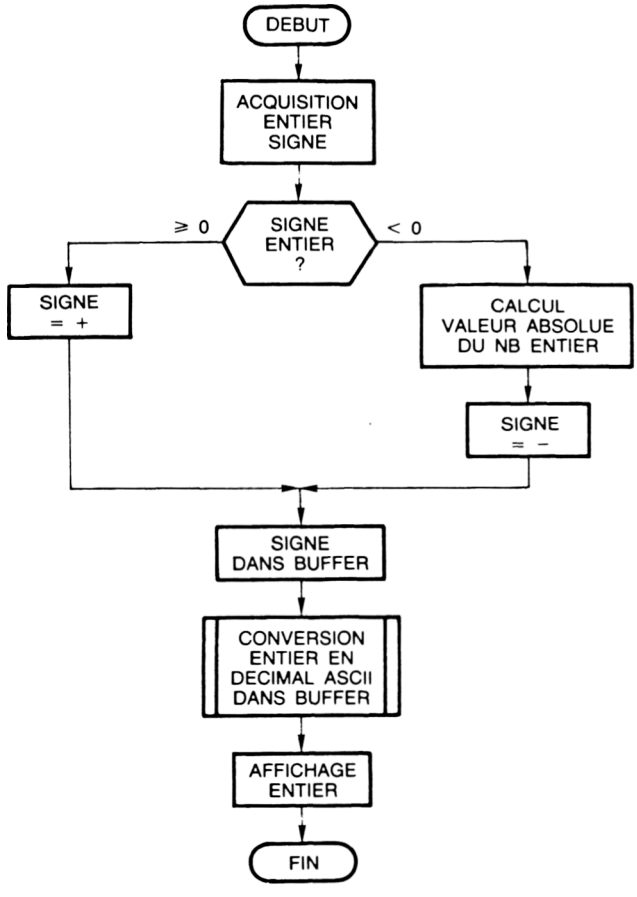
```
                 10 ;=* TRANSF ENTIER EN ASCII POUR CPC 464/664/6128 =*
                 20 ;=* SANS UTILISATION DES VECTEURS MATHEMATIQUES  =*
7000             30         ent  #7000
7000             40         org  #7000
                 50 ;--* AFFICHAGE D'UN TEXTE ASCII *--
C37D             60 affich: equ  #c37d
                 70 ;--* CONNECTION EN ROM HAUTE *--
B900             80 romhau: equ  #b900
7000  CD00B9     90         call romhau
                100 ;--* ACQUISITION ENTIER *--
7003  ED4B4370  110         ld   bc,(entier)
                120 ;--* INIT POINTEUR DE CHAINE ASCII *--
7007  DD214570  130         ld   ix,chaine
                140 ;--* NB DE FOIS 10000 COMPLET *--
700B  111027    150         ld   de,10000
700E  CD3070    160         call ascii
                170 ;--* NB DE FOIS 1000 COMPLET *--
7011  11E803    180         ld   de,1000
7014  CD3070    190         call ascii
                200 ;--* NB DE CENTAINES COMPLETES *--
7017  116400    210         ld   de,100
701A  CD3070    220         call ascii
                230 ;--* NB DE DIZAINES COMPLETES *--
701D  110A00    240         ld   de,10
7020  CD3070    250         call ascii
                260 ;--* NB D'UNITES *--
7023  110100    270         ld   de,1
7026  CD3070    280         call ascii
                290 ;--* AFFICHAGE CHAINE ASCII *--
7029  214570    300         ld   hl,chaine
702C  CD7DC3    310         call affich
702F  C9        320         ret
                330 ;--* DETERM DU NB DE CHAQUE TRANCHE *--
7030  3E2F      340 ascii:  ld   a,#2f
                350 ;--* ACQUISITION NOUVEL ENTIER *--
7032  60        360         ld   h,b
7033  69        370         ld   l,c
                380 ;--* COMPTAGE UNITES PAR TRANCHE *--
7034  3C        390 compte: inc  a
7035  ED52      400         sbc  hl,de
                410 ;--* SI POSITIF COMPTAGE UNITES *--
7037  30FB      420         jr   nc,compte
                430 ;--* PLUS D'UNITES DANS LA TRANCHE *--
                440 ;--* CORRIGE ENTIER EN POSITIF *--
7039  ED5A      450         adc  hl,de
```

```
                460 ;--* SAUVE NOUVEL ENTIER *--
703B  44        470         ld   b,h
703C  4D        480         ld   c,l
                490 ;--* CAR ASCII DANS CHAINE *--
703D  DD7700    500         ld   (ix+0),a
                510 ;--* INCREMENT POINTEUR CHAINE *--
7040  DD23      520         inc  ix
7042  C9        530         ret
                540 ;--* ZONES UTILISEES PAR LA ROUTINE *--
7043  7D00      550 entier: defw 125
7045            560 chaine: defs 6


Pass 2 errors: 00

affich C37D  ascii  7030  chaine 7045
compte 7034  entier 7043  romhau B900


Table used:    90  from   900
Executes: 28672
```

Routine de conversion et affichage d'un nombre entier signé sur 16 bits, en décimal, sans utilisation des vecteurs mathématiques

Légende :

0 → chiffre zéro



```
Hisoft GENA3.1 Assembler.


Pass 1 errors: 00


                  10 ;=* ENTIER SIGNE EN ASCII POUR CPC 464/664/6128 =*
                  20 ;=* SANS UTILISATION DES VECTEURS MATHEMATIQUES =*
7000              30         ent  #7000
7000              40         org  #7000
                  50 ;--* AFFICHAGE D'UN TEXTE ASCII *--
C37D              60 affich: equ  #c37d
                  70 ;--* CONNECTION EN ROM HAUTE *--
B900              80 romhau: equ  #b900
7000  CD00B9      90         call romhau
                 100 ;--* ACQUISITION ENTIER *--
7003  ED4B6A70   110         ld   bc,(entier)
7007  CB78       120         bit  7,b
7009  280B       130         jr   z,pos
                 140 ;--* COMPLEMENT A 2 DE ENTIER *--
700B  78         150         ld   a,b
700C  2F         160         cpl
700D  47         170         ld   b,a
700E  79         180         ld   a,c
700F  2F         190         cpl
7010  4F         200         ld   c,a
7011  03         210         inc  bc
                 220 ;--* ENTIER EST DU SGN MOINS *--
7012  3E2D       230         ld   a,#2d
7014  1802       240         jr   sgn
                 250 ;--* ENTIER EST DU SGN PLUS *--
7016  3E2B       260 pos:    ld   a,#2b
                 270 ;--* SAUVE SIGNE DEVANT CHAINE *--
7018  326C70     280 sgn:    ld   (signe),a
                 290 ;--* INIT POINTEUR DE CHAINE ASCII *--
701B  DD216D70   300         ld   ix,chaine
                 310 ;--* NB DE FOIS 10000 COMPLET *--
701F  111027     320         ld   de,10000
7022  CD5770     330         call ascii
                 340 ;--* NB DE FOIS 1000 COMPLET *--
7025  11E803     350         ld   de,1000
7028  CD5770     360         call ascii
                 370 ;--* NB DE CENTAINES COMPLETES *--
702B  116400     380         ld   de,100
702E  CD5770     390         call ascii
                 400 ;--* NB DE DIZAINES COMPLETES *--
7031  110A00     410         ld   de,10
7034  CD5770     420         call ascii
                 430 ;--* NB D'UNITES *--
7037  110100     440         ld   de,1
```

```
703A  CD5770    450          call ascii
                460 ;--* SUP ZEROS NON SIGNIF DANS CHAINE *--
703D  216D70    470          ld   hl,chaine
7040  2B        480          dec  hl
7041  010700    490          ld   bc,7
7044  0B        500 bcl:     dec  bc
7045  23        510          inc  hl
7046  7E        520          ld   a,(hl)
7047  FE30      530          cp   #30
7049  28F9      540          jr   z,bcl
                550 ;--* TRANSF CHIFFR SIGNIF AU DEBUT *--
704B  116D70    560          ld   de,chaine
704E  EDB0      570          ldir
                580 ;--* AFFICHAGE CHAINE ASCII *--
7050  216C70    590          ld   hl,signe
7053  CD7DC3    600          call affich
7056  C9        610          ret
                620 ;--* DETERM DU NB DE CHAQUE TRANCHE *--
7057  3E2F      630 ascii:   ld   a,#2f
                640 ;--* ACQUISITION NOUVEL ENTIER *--
7059  60        650          ld   h,b
705A  69        660          ld   l,c
                670 ;--* COMPTAGE UNITES PAR TRANCHE *--
705B  3C        680 compte:  inc  a
705C  ED52      690          sbc  hl,de
                700 ;--* SI POSITIF COMPTAGE UNITES *--
705E  30FB      710          jr   nc,compte
                720 ;--* PLUS D'UNITES DANS LA TRANCHE *--
                730 ;--* CORRIGE ENTIER EN POSITIF *--
7060  ED5A      740          adc  hl,de
                750 ;--* SAUVE NOUVEL ENTIER *--
7062  44        760          ld   b,h
7063  4D        770          ld   c,l
                780 ;--* CAR ASCII DANS CHAINE *--
7064  DD7700    790          ld   (ix+0),a
                800 ;--* INCREMENT POINTEUR CHAINE *--
7067  DD23      810          inc  ix
7069  C9        820          ret
                830 ;--* ZONES UTILISEES PAR LA ROUTINE *--
706A  D5DD      840 entier:  defw #ddd5
706C            850 signe:   defs 1
706D            860 chaine:  defs 6
```

```
Pass 2 errors: 00

affich C37D  ascii  7057  bcl    7044
chaine 706D  compte 705B  entier 706A
pos    7016  romhau B900  sgn    7018
signe  706C

Table used:   132  from   900
Executes: 28672
```

ROUTINES DE TRAITEMENT DES NOMBRES FLOTTANTS

Rappels

Un nombre flottant permet de représenter un nombre réel, positif ou négatif, en mémoire.
Un nombre réel est constitué d'un signe, d'une partie entière, et d'une partie décimale.
Pour représenter un nombre réel en mémoire, un nombre flottant est constitué d'une mantisse et d'un exposant.

La mantisse représente une chaîne de bits, intégrant le signe, la partie entière, et la partie décimale du nombre réel ; mais rien ne permet de distinguer la partie entière de la partie décimale.

L'exposant quant à lui permet de pallier cette lacune au niveau de la mantisse, puisqu'en effet c'est lui qui permet d'indiquer la séparation entre la partie entière et la partie décimale.

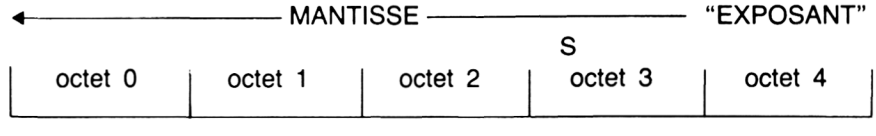
Représentation d'un flottant sur Amstrad CPC :

Cinq octets suffisent pour représenter n'importe quel nombre réel en mémoire de l'Amstrad CPC.
Toutefois, il ne faut pas perdre de vue, le fait que plus un réel est grand, plus la précision obtenue diminue, et ceci est encore plus vrai pour les décimales.

Dans le cas de l'Amstrad CPC, la configuration des cinq octets d'un flottant est la suivante :

— MANTISSE → 4 premiers octets (octets 0 à 3)
— EXPOSANT → cinquième octet (octet 4)

Le signe du nombre réel est déterminé par le bit 7 de l'octet 3. La représentation de notre nombre flottant sera donc la suivante :



La lecture de la mantisse se fait de l'octet 3 vers l'octet 0.

Pour déterminer la frontière de bit entre la partie entière et la partie décimale, il nous faut soustraire #80 de l'octet 4. Le nombre ainsi obtenu nous donne le nombre de bits à prendre, en partant de l'octet 3 vers l'octet 0, pour la partie entière du nombre flottant.

Une fois le signe obtenu, et avant tout traitement, le bit 7 de l'octet 3 est mis systématiquement à 1 ; en effet, à la création d'un nombre flottant, la partie entière commence obligatoirement par un bit à 1 ; mais celui-ci est mis à 0 ou à 1 pour distinguer un nombre réel positif d'un nombre réel négatif.

Lecture d'un nombre flottant

Dès que nous avons déterminé le nombre de bits à traiter pour la partie entière, nous disposerons les octets 3 à 0 de la droite vers la gauche, pour procéder à la lecture de la partie entière et de la partie décimale du nombre flottant.

Nous allons raisonner sur un exemple que nous conserverons pour la suite du traitement d'un nombre flottant.

Soit le nombre réel positif : 1235.786

Sa représentation mémoire sera :

HEXA	→ DD	24	E7	40	8E
OCTETS	→ 0	1	2	3	4

En binaire, et en partant de l'octet 3, la mantisse sera :

7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
0 1 0 0 0 0 0 0	1 1 1 0 0 1 1 1	0 0 1 0 0 1 0 0	1 1 0 1 1 1 0 1
OCTET 3	OCTET 2	OCTET 1	OCTET 0

L'exposant étant #8E, le nombre de bits à traiter pour la partie entière sera 14 (ou E en hexadécimal).

Avant tout traitement, et après avoir remarqué que le bit 7 de l'octet 3 est à 0, soit un nombre positif, nous devons mettre le bit 7 de cet octet à 1.

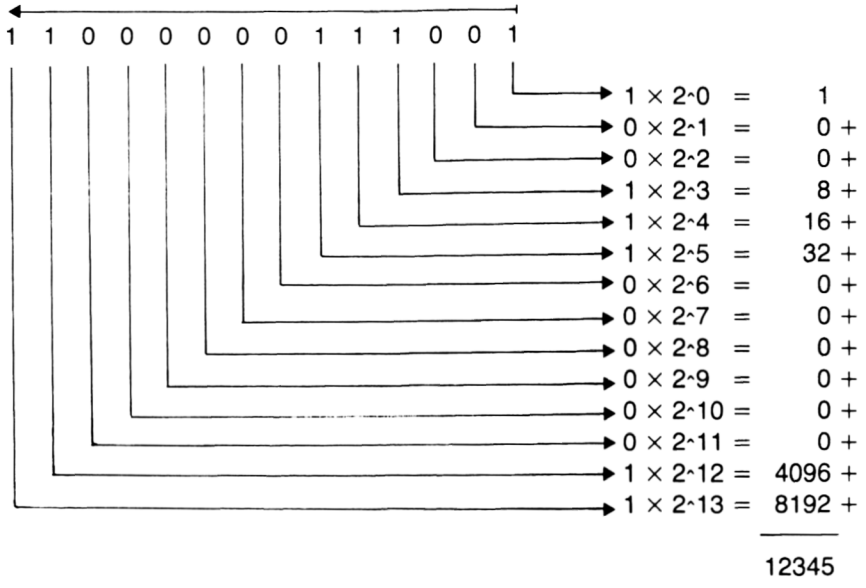
Le schéma suivant va nous indiquer la position virtuelle de la virgule flottante, ainsi que les sens de lecture binaire de la partie entière et de la lecture de la partie décimale. Le nombre de l'exposant indique la position du 1er bit de la partie entière ; celui-ci, repéré par le nombre #E, indique que son « poids » sera de 1, soit 2 puissance 0. La lecture binaire devra donc s'effectuer en partant de ce bit, soit du bit de « poids faible » au bit de « poids fort » (2 à la puissance 13).

←——14 premiers bits——→ (exposant = #8E)

1 2 3 4 5 6 7 8	9 A B C D E		
1 1 0 0 0 0 0 0	1 1 1 0 0 1 1 1	0 0 1 0 0 1 0 0	1 1 0 1 1 1 0 1
OCTET 3	OCTET 2	OCTET 1	OCTET 0

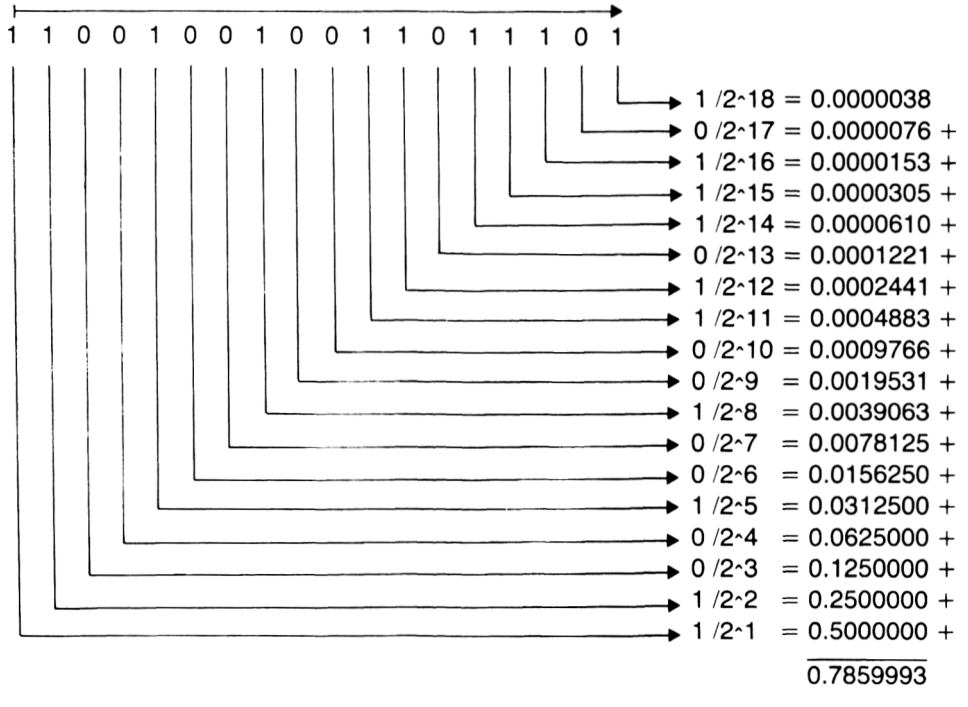
←——— lecture partie entière ———|——— lecture partie décimale ———→

a) calcul de la partie entière :



Le principe est très simple, puisqu'il s'agit de multiplier chaque bit par une puissance de 2 correspondant au poids du bit traité. Il suffit ensuite de sommer chaque nombre ainsi obtenu pour obtenir la partie entière, soit ici 12345.
Les poids des bits vont de 0 à 13, en partant du bit le plus à droite des 14 bits.

b) calcul de la partie décimale :



Le principe est presque aussi simple que pour la partie entière, puisqu'il s'agit de diviser chaque bit par une puissance de 2 correspondant au poids du bit traité.

Il suffit ensuite de sommer chaque nombre ainsi obtenu pour obtenir la partie décimale, soit ici 0.7859993.

Les puissances des bits vont de 1 à 18, en partant du bit le plus à gauche des 18 bits de la partie décimale.

Remarque importante

On constatera que la partie décimale obtenue ne nous donne pas exactement le résultat désiré.

Si nous voulons une approximation au 9/1000e, le traitement des 6 premiers bits de la partie décimale suffit amplement ; dans ce cas, seuls les 3 premiers chiffres décimaux seront conservés. C'est ce nombre de bits que nous traiterons dans la routine.

ROUTINE D'AFFICHAGE D'UN FLOTTANT, EN DÉCIMAL

La méthode utilisée est une méthode classique qui consiste à isoler la partie entière, puis la partie décimale, du flottant.

Prenons un exemple pour bien comprendre le processus utilisé. Soit le nombre flottant négatif : −123.45.

Première étape :

Déterminer le signe du nombre flottant traité, en testant le bit 7 de l'octet 3, et le conserver pour l'affichage ultérieur.
Puis mettre le bit 7 de l'octet 3 à 1 pour traiter le nombre en valeur absolue.

Deuxième étape :

Isoler la partie entière de la valeur absolue du flottant traité.
Soit : ZENT = INT (123.45) = 123 (à conserver).

Troisième étape :

Isoler la partie décimale de la valeur absolue du flottant.
Soit : ZDEC = 123.45 − ZENT = 0.45 (à conserver).

Quatrième étape :

Mettre la partie décimale sous la forme d'une valeur entière.
Soit : ZDEC = ZDEC × 1000 = 450 (obtention de 3 décimales).

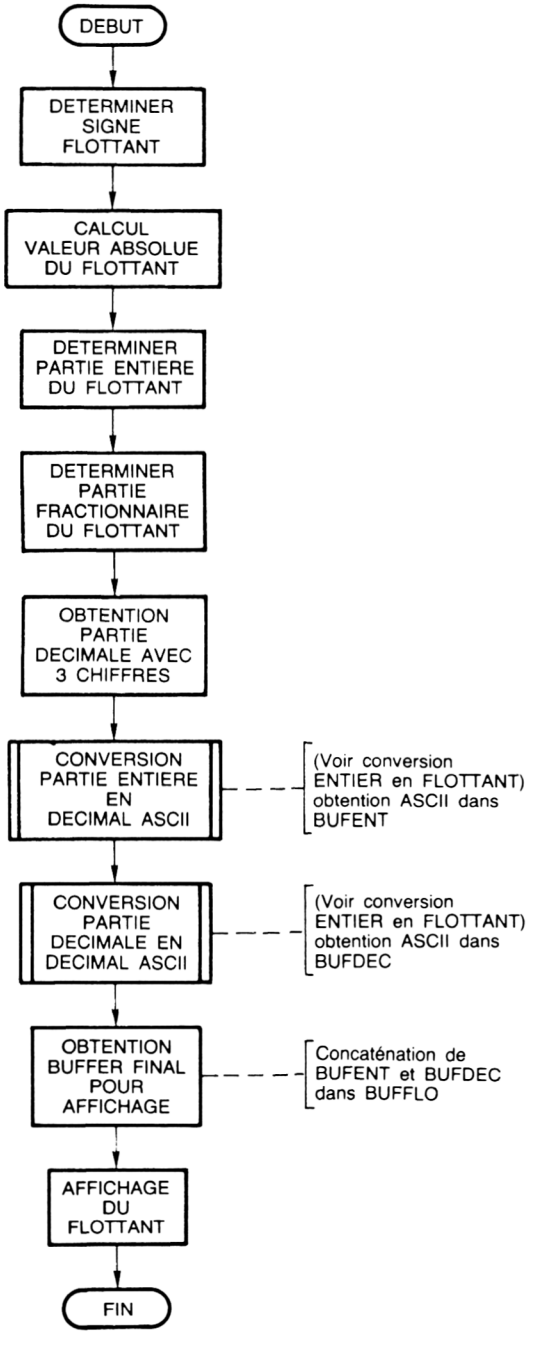
Cinquième étape :

Rassembler le signe, la partie entière, la virgule, et la partie décimale dans un buffer ASCII, pour affichage.
Soit : BUFFLO = −123,450.

Les diverses étapes exposées seront réalisées simplement, par utilisation des routines mathématiques flottantes en RAM.

Ci-après, vous trouverez l'organigramme de programmation, ainsi que la routine assembleur correspondante.

Routine de conversion et affichage d'un flottant, en décimal, avec utilisation des vecteurs mathématiques en RAM



```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                  10 ;=* ROUT AFF VALEURS FLOTTANTES *=
                  20 ;=* POUR CPC 6128 *=
7000              30          ent  #7000
7000              40          org  #7000
                  50 ;--* CONNECTION EN ROM HAUTE *--
B900              60 romhau: equ  #b900
                  70 ;--* TRANS. ENTIER EN FLOTTANT *--
BD64              80 entflo: equ  #bd64
                  90 ;--* OPERATIONS EN FLOTTANT *--
BD88             100 divise: equ  #bd88
BD85             110 multip: equ  #bd85
BD82             120 soustr: equ  #bd82
BD7C             130 additi: equ  #bd7c
                 140 ;--* RECUP PARTIE ENTIERE D'UN FLOTTANT *--
BD73             150 intflo: equ  #bd73
                 160 ;--* TRANS. FLOTTANT EN ENTIER *--
BD6A             170 floent: equ  #bd6a
                 180 ;--* TEST SIGNE FLOTTANT *--
BD94             190 sgnflo: equ  #bd94
                 200 ;--* NEGATION D'UN FLOTTANT *--
BD91             210 negflo: equ  #bd91
                 220 ;--* ROUTINE D'AFFICHAGE D'UN TEXTE ASCII *--
C37D             230 affich: equ  #c37d
                 240 ;
7000  CD00B9     250          call romhau
                 260 ;--* CHARG EX DE VAL FLOTTANT DANS ZFLOT *--
7003  DD211372   270          ld   ix,zflot
7007  3E00       280          ld   a,0
7009  DD7700     290          ld   (ix+0),a
700C  3E00       300          ld   a,0
700E  DD7701     310          ld   (ix+1),a
7011  3E00       320          ld   a,0
7013  DD7702     330          ld   (ix+2),a
7016  3EA8       340          ld   a,#a8
7018  DD7703     350          ld   (ix+3),a
701B  3E84       360          ld   a,#84
701D  DD7704     370          ld   (ix+4),a
                 380 ;--* SAUVE SIGNE DU FLOTTANT *--
7020  DD211372   390          ld   ix,zflot
7024  DD7E03     400          ld   a,(ix+3)
7027  CB7F       410          bit  7,a
7029  2004       420          jr   nz,moins
702B  3E2B       430          ld   a,#2b
702D  1802       440          jr   sgn
702F  3E2D       450 moins:   ld   a,#2d
```

```
7031  325672     460 sgn:    ld   (signe),a
                 470 ;--* CALCUL VAL ABS DU FLOTTANT *--
7034  DD7E03     480         ld   a,(ix+3)
7037  CB27       490         sla  a
7039  CB2F       500         sra  a
703B  DD7703     510         ld   (ix+3),a
                 520 ;--* ZFLOT DANS ACCUM1 *--
703E  211372     530         ld   hl,zflot
7041  114072     540         ld   de,accum1
7044  010500     550         ld   bc,5
7047  EDB0       560         ldir
                 570 ;--* RECUP  PARTIE ENTIERE DE ACCUM1 *--
7049  214072     580         ld   hl,accum1
704C  CD73BD     590         call intflo
704F  2A4072     600         ld   hl,(accum1)
                 610 ;--* HL DANS ZENT *--
7052  221872     620         ld   (zent),hl
                 630 ;--* ZENT EN FLOTTANT DANS ACCUM1 *--
7055  2A1872     640         ld   hl,(zent)
7058  114072     650         ld   de,accum1
705B  CD64BD     660         call entflo
705E  DD214072   670         ld   ix,accum1
7062  DD7E03     680         ld   a,(ix+3)
7065  CB27       690         sla  a
7067  CB2F       700         sra  a
7069  DD7703     710         ld   (ix+3),a
                 720 ;--* ACCUM1 DANS ACCUM2 *--
706C  214072     730         ld   hl,accum1
706F  114572     740         ld   de,accum2
7072  010500     750         ld   bc,5
7075  EDB0       760         ldir
                 770 ;--* ZFLOT DANS ACCUM1 *--
7077  211372     780         ld   hl,zflot
707A  114072     790         ld   de,accum1
707D  010500     800         ld   bc,5
7080  EDB0       810         ldir
                 820 ;--* ACCUM1 = ACCUM1 - ACCUM2 *--
7082  214072     830         ld   hl,accum1
7085  114572     840         ld   de,accum2
7088  CD82BD     850         call soustr
                 860 ;--* 1000 DANS ACCUM2 *--
708B  2A4A72     870         ld   hl,(mille)
708E  114572     880         ld   de,accum2
7091  CD64BD     890         call entflo
                 900 ;--* ACCUM1 = ACCUM1 * 1000 *--
7094  214072     910         ld   hl,accum1
7097  114572     920         ld   de,accum2
709A  CD85BD     930         call multip
                 940 ;--* FLOTTANT EN ENTIER DANS HL *--
```

```
709D  214072   950          ld   hl,accuml
70A0  CD73BD   960          call intflo
70A3  2A4072   970          ld   hl,(accuml)
               980 ;--* HL DANS ZDEC *--
70A6  221A72   990          ld   (zdec),hl
              1000 ;--* ZENT DANS ENTIER *--
70A9  2A1872  1010          ld   hl,(zent)
70AC  225472  1020          ld   (entier),hl
              1030 ;--* ENTIER EN ASCII DANS BUFFER *--
70AF  CD1B71  1040          call ascii
              1050 ;--* BUFFER DANS BUFENT *--
70B2  211C72  1060          ld   hl,buffer
70B5  112372  1070          ld   de,bufent
70B8  010600  1080          ld   bc,6
70BB  EDB0    1090          ldir
              1100 ;--* ZDEC DANS ENTIER *--
70BD  2A1A72  1110          ld   hl,(zdec)
70C0  225472  1120          ld   (entier),hl
              1130 ;--* ENTIER EN ASCII DANS BUFFER *--
70C3  CD1B71  1140          call ascii
              1150 ;--* BUFFER DANS BUFDEC *--
70C6  211C72  1160          ld   hl,buffer
70C9  112972  1170          ld   de,bufdec
70CC  010600  1180          ld   bc,6
70CF  EDB0    1190          ldir
              1200 ;--* BUFENT DANS BUFFLO SAUF BLANCS *--
70D1  010600  1210          ld   bc,6
70D4  212372  1220          ld   hl,bufent
70D7  112F72  1230          ld   de,bufflo
70DA  7E      1240 sib:     ld   a,(hl)
70DB  FE20    1250          cp   #20
70DD  2004    1260          jr   nz,tbe
70DF  23      1270          inc  hl
70E0  0D      1280          dec  c
70E1  18F7    1290          jr   sib
70E3  C5      1300 tbe:     push bc
70E4  EDB0    1310          ldir
70E6  C1      1320          pop  bc
              1330 ;--* BUFDEC DANS BUFFLO SAUF BLANCS ET SGN *--
70E7  212F72  1340          ld   hl,bufflo
70EA  09      1350          add  hl,bc
70EB  3E2C    1360          ld   a,#2c
70ED  77      1370          ld   (hl),a
70EE  23      1380          inc  hl
70EF  54      1390          ld   d,h
70F0  5D      1400          ld   e,l
70F1  212972  1410          ld   hl,bufdec
70F4  010600  1420          ld   bc,6
70F7  7E      1430 sibmp:   ld   a,(hl)
```

```
70F8  FE20      1440          cp   #20
70FA  2002      1450          jr   nz,sim
70FC  180A      1460          jr   ptb
70FE  FE2D      1470 sim:     cp   #2d
7100  2002      1480          jr   nz,sip
7102  1804      1490          jr   ptb
7104  FE2B      1500 sip:     cp   #2b
7106  2004      1510          jr   nz,tbel
7108  23        1520 ptb:     inc  hl
7109  0D        1530          dec  c
710A  18EB      1540          jr   sibmp
710C  EDB0      1550 tbel:    ldir
                1560 ;--* AFFICHAGE BUFFLO *--
710E  3A5672    1570          ld   a,(signe)
7111  322F72    1580          ld   (bufflo),a
7114  212F72    1590          ld   hl,bufflo
7117  CD7DC3    1600          call affich
711A  C9        1610          ret
                1620 ;--* INITIALISATION BUFFER A BLANC *--
711B  3E06      1630 ascii:   ld   a,6
711D  DD211C72  1640          ld   ix,buffer
7121  DD360020  1650 blbuf:   ld   (ix+0),#20
7125  DD23      1660          inc  ix
7127  3D        1670          dec  a
7128  20F7      1680          jr   nz,blbuf
                1690 ;--* INITIALISATION POINTEUR BUFFER *--
712A  012272    1700          ld   bc,finbuf
712D  ED435272  1710          ld   (pb),bc
                1720 ;--* INIT. FIN BUFFER A ZERO POUR ROUTINE AFF. *--
7131  3E00      1730          ld   a,0
7133  322272    1740          ld   (finbuf),a
                1750 ;--* SAUVE SIGNE DE ENTIER *--
7136  2A5472    1760          ld   hl,(entier)
7139  E5        1770          push hl
                1780 ;--* TRAIT. VAL. ABS ENTIER ORIGINAL *--
713A  3E0A      1790 chiffr:  ld   a,10
713C  DD214072  1800          ld   ix,accuml
7140  DD360000  1810 mazacc:  ld   (ix+0),0
7144  DD23      1820          inc  ix
7146  3D        1830          dec  a
7147  20F7      1840          jr   nz,mazacc
                1850 ;--* RECUP. ENTIER EN FLOTTANT *--
7149  2A5472    1860          ld   hl,(entier)
714C  114072    1870          ld   de,accuml
714F  CD64BD    1880          call entflo
                1890 ;--* SAUVE FLOTTANT EN STO *--
7152  214072    1900          ld   hl,accuml
7155  113B72    1910          ld   de,sto
7158  010500    1920          ld   bc,5
```

```
715B  EDB0      1930          ldir
                1940 ;--* 10 DANS ACCUM2 *--
715D  2A4E72    1950          ld   hl,(dix)
7160  114572    1960          ld   de,accum2
7163  CD64BD    1970          call entflo
                1980 ;--* ACCUM1 = ACCUM1 / 10 *--
7166  214072    1990          ld   hl,accum1
7169  114572    2000          ld   de,accum2
716C  CD88BD    2010          call divise
                2020 ;--* RECUP PARTIE ENTIERE DE ACCUM1 *--
716F  214072    2030          ld   hl,accum1
7172  CD73BD    2040          call intflo
                2050 ;--* SAUVE PARTIE ENTIERE DANS STOENT *--
7175  ED4B4072  2060          ld   bc,(accum1)
7179  ED435072  2070          ld   (stoent),bc
                2080 ;--* TRANS. ENTIER STOENT EN FLOT. DANS ACCUM1 *--
717D  2A5072    2090          ld   hl,(stoent)
7180  114072    2100          ld   de,accum1
7183  CD64BD    2110          call entflo
                2120 ;--* 10 DANS ACCUM2 *--
7186  2A4E72    2130          ld   hl,(dix)
7189  114572    2140          ld   de,accum2
718C  CD64BD    2150          call entflo
718F  1802      2160          jr   suite
                2170 ;--* PONT POUR REMONTEE A CHIFFR (RELOGEABLE) *--
7191  18A7      2180 ch:      jr   chiffr
                2190 ;--* ACCUM1 = ACCUM1 * 10 *--
7193  114572    2200 suite:   ld   de,accum2
7196  214072    2210          ld   hl,accum1
7199  CD85BD    2220          call multip
                2230 ;--* ACCUM1 DANS ACCUM2 *--
719C  214072    2240          ld   hl,accum1
719F  114572    2250          ld   de,accum2
71A2  010500    2260          ld   bc,5
71A5  EDB0      2270          ldir
                2280 ;--* STO DANS ACCUM1 *--
71A7  213B72    2290          ld   hl,sto
71AA  114072    2300          ld   de,accum1
71AD  010500    2310          ld   bc,5
71B0  EDB0      2320          ldir
                2330 ;--* TEST ET BRANCH. SI EXP. ACCUM2 A ZERO *--
71B2  DD214572  2340          ld   ix,accum2
71B6  DD7E04    2350          ld   a,(ix+4)
71B9  FE00      2360          cp   0
71BB  CC0E72    2370          call z,exp80
                2380 ;--* ACCUM1 = ACCUM1 - ACCUM2 *--
71BE  214072    2390          ld   hl,accum1
71C1  114572    2400          ld   de,accum2
71C4  CD82BD    2410          call soustr
```

```
                2420 ;--* #30 DANS ACCUM2 *--
71C7  2A4C72    2430         ld   hl,(trente)
71CA  114572    2440         ld   de,accum2
71CD  CD64BD    2450         call entflo
                2460 ;--* ACCUM1 = ACCUM1 + ACCUM2 (TRANS. ASCII) *--
71D0  214072    2470         ld   hl,accum1
71D3  114572    2480         ld   de,accum2
71D6  CD7CBD    2490         call additi
                2500 ;--* RECUP ASCII DANS HL *--
71D9  214072    2510         ld   hl,accum1
71DC  CD73BD    2520         call intflo
71DF  2A4072    2530         ld   hl,(accum1)
                2540 ;--* POINT BUF = POINT BUF - 1 *--
71E2  ED4B5272  2550         ld   bc,(pb)
71E6  0B        2560         dec  bc
71E7  ED435272  2570         ld   (pb),bc
                2580 ;--* TRANS. ASCII DANS BUFFER *--
71EB  7D        2590         ld   a,l
71EC  2A5272    2600         ld   hl,(pb)
71EF  77        2610         ld   (hl),a
                2620 ;--* TEST ET BRANCH. SI STOENT = 0 *--
71F0  2A5072    2630         ld   hl,(stoent)
71F3  7D        2640         ld   a,l
71F4  FE00      2650         cp   0
71F6  2805      2660         jr   z,affbuf
                2670 ;--* SINON ENTIER = ASCII (HL) *--
71F8  225472    2680         ld   (entier),hl
71FB  1894      2690         jr   ch
                2700 ;--* TEST SIGNE ENTIER ORIGINAL *--
71FD  E1        2710 affbuf: pop  hl
71FE  CB7C      2720         bit  7,h
7200  2804      2730         jr   z,plus
                2740 ;--* SIGNE NEGATIF DANS BUFFER *--
7202  3E2D      2750         ld   a,#2d
7204  1802      2760         jr   sorbuf
                2770 ;--* SIGNE POSITIF DANS BUFFER *--
7206  3E2B      2780 plus:   ld   a,#2b
7208  2A5272    2790 sorbuf: ld   hl,(pb)
720B  2D        2800         dec  l
720C  77        2810         ld   (hl),a
720D  C9        2820         ret
                2830 ;--* CORRECTION EXPOSANT A #80 *--
720E  DD360480  2840 exp80:  ld   (ix+4),#80
7212  C9        2850         ret
                2860 ;--* ZONES UTILISEES PAR LA ROUTINE *--
7213            2870 zflot:  defs 5                    ;RECEPTION FLOT
7218            2880 zent:   defs 2                    ;PART ENT DU FLOT
721A            2890 zdec:   defs 2                    ;PART DEC DU FLOT
721C            2900 buffer: defs 6                    ;BUFFER (dont SGN)
```

```
7222            2910 finbuf: defs 1          ;FIN BUFFER
7223            2920 bufent: defs 6          ;BUF ASCII PART ENT
7229            2930 bufdec: defs 6          ;BUF ASCII PART DEC
722F            2940 bufflo: defs 12         ;BUF ASCII DU FLOT
723B            2950 sto:    defs 5          ;STOCKAGE FLOTTANT
7240            2960 accum1: defs 5          ;ZONE DE CALCUL
7245            2970 accum2: defs 5          ;ZONE DE CALCUL
724A  E803      2980 mille:  defw 1000
724C  3000      2990 trente: defw #30
724E  0A00      3000 dix:    defw 10
7250            3010 stoent: defs 2          ;STOCKAGE ENTIER
7252            3020 pb:     defs 2          ;POINTEUR BUFFER
7254            3030 entier: defs 2          ;PARAM POUR TRAIT
7256            3040 signe:  defs 1          ;SIGNE DU FLOTTANT


Pass 2 errors: 00

accum1 7240  accum2 7245  additi BD7C
affbuf 71FD  affich C37D  ascii  711B
blbuf  7121  bufdec 7229  bufent 7223
buffer 721C  bufflo 722F  ch     7191
chiffr 713A  divise BD88  dix    724E
entflo BD64  entier 7254  exp80  720E
finbuf 7222  floent BD6A  intflo BD73
mazacc 7140  mille  724A  moins  702F
multip BD85  negflo BD91  pb     7252
plus   7206  ptb    7108  romhau B900
sgn    7031  sgnflo BD94  sib    70DA
sibmp  70F7  signe  7256  sim    70FE
sip    7104  sorbuf 7208  soustr BD82
sto    723B  stoent 7250  suite  7193
tbe    70E3  tbel   710C  trente 724C
zdec   721A  zent   7218  zflot  7213


Table used:   588  from   900
Executes: 28672
```

ROUTINE DE RÉCUPÉRATION DE LA PARTIE ENTIÈRE D'UN FLOTTANT

Un entier 16 bits signé permet d'obtenir des valeurs comprises entre −32768 et +32767.
A partir de ce rappel, nous pouvons dire que seuls les 15 premiers bits de la mantisse seront nécessaires pour récupérer la partie entière d'un nombre flottant.

Le processus de traitement de la partie entière, précédemment énoncé, n'est pas applicable directement, en raison de l'absence d'instructions Z80 permettant l'accès indicé à un bit.

Il nous faut donc adopter un autre processus, qui lui fait appel à des instructions de décalage, de test d'un bit donné, et de positionnement de bit à 0 ou à 1.

A partir de notre exemple de départ, nous allons expliquer en détails le fonctionnement de cette méthode, qui n'est certes pas la seule existante en matière de traitement d'un flottant...

Soit le flottant pris en exemple :	1235.786 →	DD	24	E7	40	8E	(H)
		0	1	2	3	4	

Afin de se rapprocher le plus possible de la routine présentée ultérieurement, pour une meilleure compréhension de celle-ci, nous ferons l'analyse du processus en nous servant des doubles registres utilisés dans la routine en question.

Première étape :

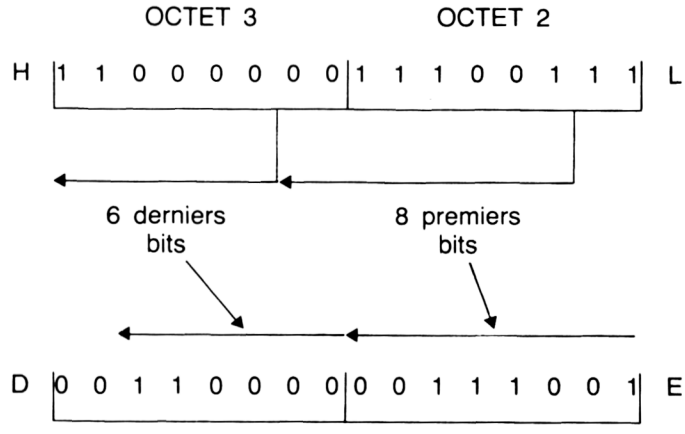
Mettre les octets 3 et 2 dans le double registre HL :

octet 3 dans l'octet fort, soit dans H,
octet 2 dans l'octet faible, soit dans L.

Deuxième étape

Nous récupérerons la partie entière dans le double registre DE. Dans cette seconde étape, il s'agira de remplir l'octet faible de ce double registre (soit E) avec les 8 premiers bits de la partie entière (voir lecture de la partie entière).

Schématiquement, nous devons, après traitement des étapes 2 et 3, obtenir ceci :



La récupération des 8 premiers bits n'est pas si simple, car 6 bits se trouvent sur les bits 2 à 7 de L, et les 2 derniers bits se trouvent sur les bits 0 et 1 de H.

La récupération de chacun des 8 bits se fera à partir du bit 0 de chacun des octets, soit respectivement sur L, puis sur H.

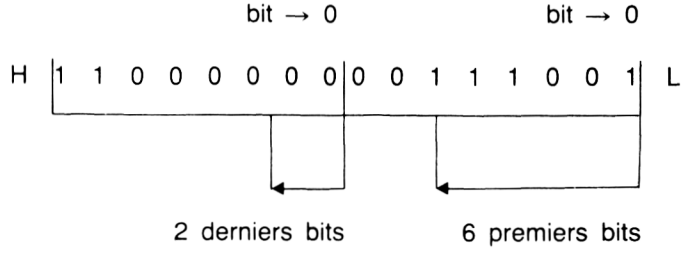
Il nous faut donc, au départ, positionner les 6 premiers bits de façon à obtenir le premier des 6 bits sur le bit 0 de L ; pour cela, quelques décalages à droite suffiront.

Le nombre de décalages à droite s'obtiendra aisément par le calcul suivant :

nb de décalages à droite = 16 − nb de bits de la partie entière

soit dans notre exemple : 16 − 14 = 2 décalages à droite.

Après décalage de 2 bits vers la droite, sur L, nous obtenons le double registre HL suivant :



Chacun des 6 bits de L sera testé en bit 0 de L, et suivant le résultat, nous placerons un bit 0 ou 1 dans le bit 7 de E.

Après chaque bit traité, un décalage à droite sur L devra être fait pour amener le bit suivant en bit 0.

De même, un décalage à droite sur E sera également nécessaire, afin de réceptionner le bit suivant, sans détruire celui qui a été acquis précédemment.

Il en sera de même pour les 2 bits restant à traiter, et qui se trouvent sur le registre H.

Schématiquement, nous aurons 8 étapes successives et similaires. → VOIR LE SCHEMA DE TRAITEMENT DES 8 ETAPES : PAGE SUIVANTE ←

Après obtention de l'octet faible du registre double DE, nous devons obtenir la configuration suivante pour HL :

H	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	L

Troisième étape :

Souvenez-vous du schéma de traitement de départ ; nous devons obtenir maintenant les 6 derniers bits, se trouvant sur H, dans l'octet fort de DE, soit en E.

Or nous constatons, avec une certaine satisfaction, que les décalages effectués sur le registre H ont permis de positionner l'ensemble des 6 bits restants à partir du bit 0 de H.

Il nous suffit donc d'effectuer le transfert du registre H, dans le registre D, pour enfin obtenir notre partie entière dans DE.

Après ce dernier transfert, nous obtenons le registre DE suivant :

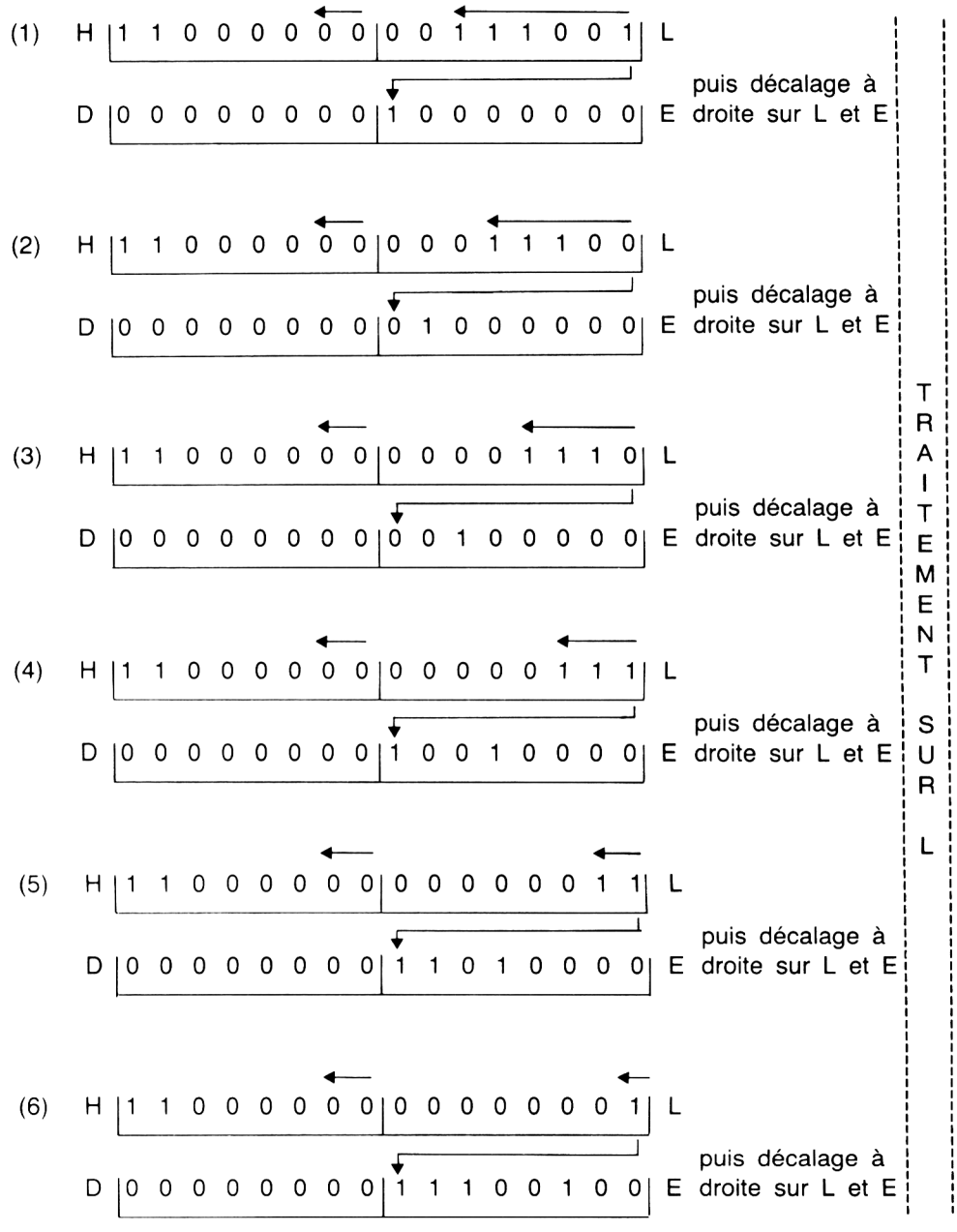
D	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	E
							#30						#39				

Soit la valeur hexa #3039, qui correspond bien à 12345 en décimal :

L'exemple présenté constitue le cas le plus complexe, soit pour un exposant supérieur à 8.

Nous allons voir ci-après le cas le plus simple, soit pour un exposant inférieur ou égal à 8.

Schéma de traitement des 8 étapes



```
(7)  H |1 1 0 0 0 0 0 0|0 0 0 0 0 0 0 0| L
       |_______________|_______________|
                       |
                       v
                                          puis décalage à
     D |0 0 0 0 0 0 0 0|0 1 1 1 0 0 1 0| E droite sur H et E
       |_______________|_______________|

(8)  H |0 1 1 0 0 0 0 0|0 0 0 0 0 0 0 0| L
       |_______________|_______________|
                       |
                       v
                                          puis décalage à
     D |0 0 0 0 0 0 0 0|0 0 1 1 1 0 0 1| E droite sur H
       |_______________|_______________|
```

TRAITEMENT H

Voyons maintenant le cas d'un flottant qui aurait un exposant inférieur ou égal à 8.

Dans ce cas, seul l'octet 3 est nécessaire pour le traitement. La même méthode que précédemment sera appliquée, mais uniquement sur 1 octet.

Ici également, nous raisonnerons en fonction de la routine qui est présentée ci-après. Nous utiliserons pour ce faire le registre double BC, dans lequel nous récupérerons la partie entière du flottant.
En fait, seul le registre C sera utilisé, puisque nous ne traiterons qu'un seul octet.

Prenons un exemple pour bien fixer les idées ; soit le réel 10, qui, en flottant, donne la configuration suivante : 00 00 00 20 84
0 1 2 3 4

Première étape :

Calcul du décalage à droite à effectuer sur le registre C.

nb de décalage = 8 – (exposant – #80)

soit dans notre exemple : nb de décalages = 8 – 4 = 4

N'oublions pas de mettre 1 dans le bit 7 du registre C, avant d'effectuer les 4 décalages à droite sur celui-ci.

Nous avons alors la configuration suivante pour le registre C :

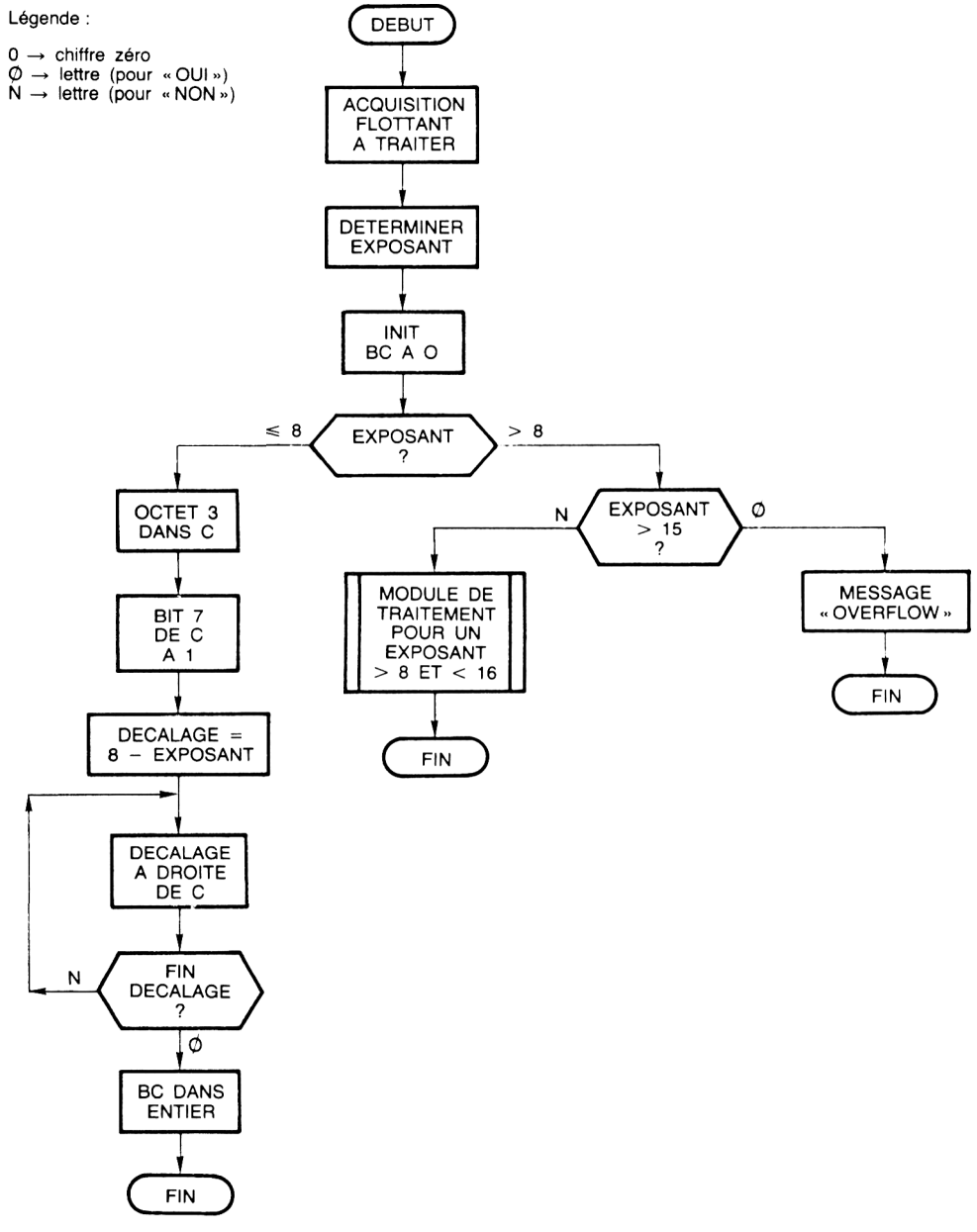
```
bit → 7 6 5 4 3 2 1 0
    C |1 0 1 0 0 0 0 0|
      |_________________|
```

Après avoir effectué les 4 décalages à droite, nous obtenons :

bit →	7	6	5	4	3	2	1	0
C	0	0	0	0	1	0	1	0

Soit la valeur hexa #0A, qui correspond bien à 10 en décimal.

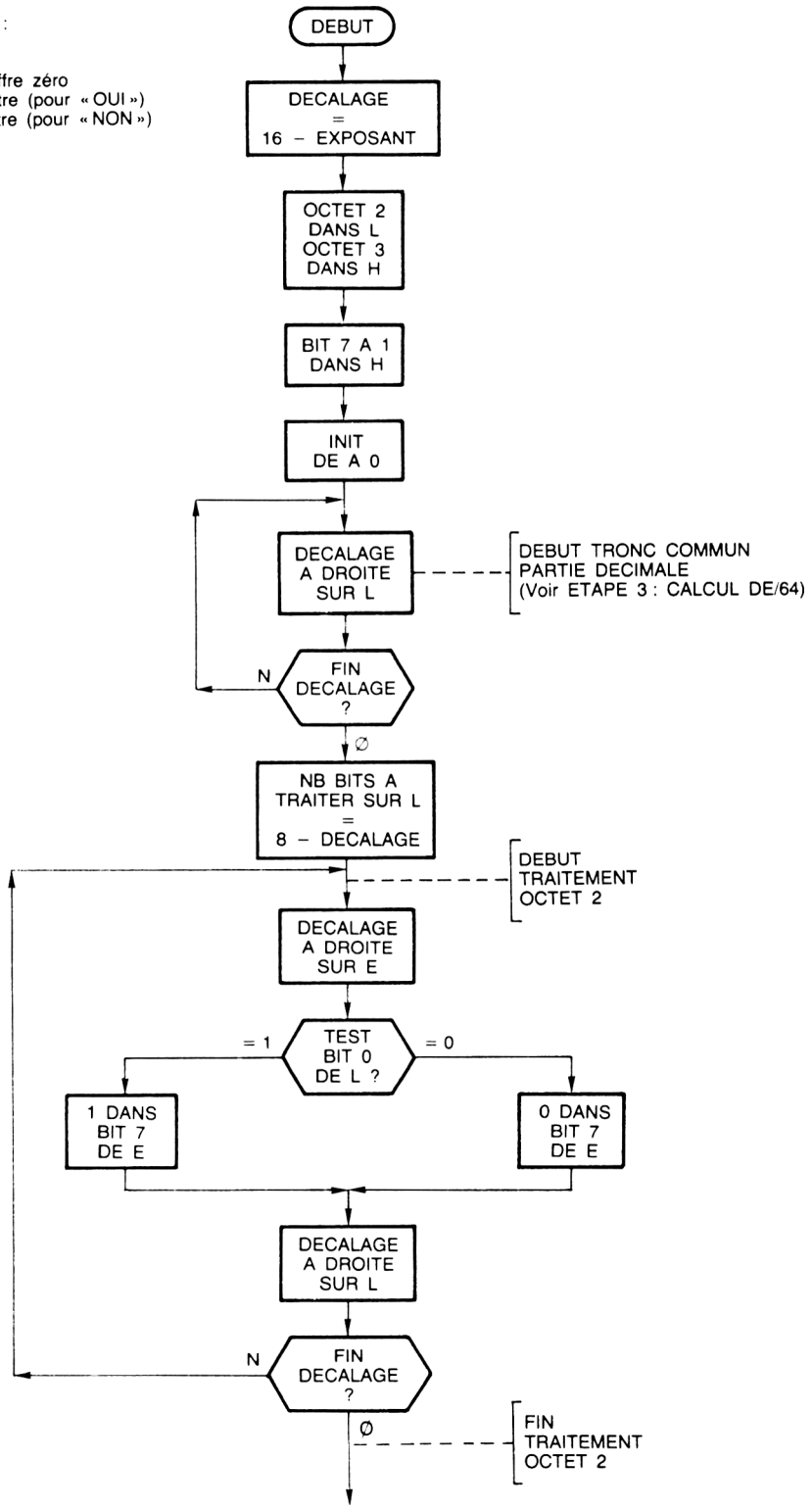
Routine de récupération de la partie entière d'un flottant



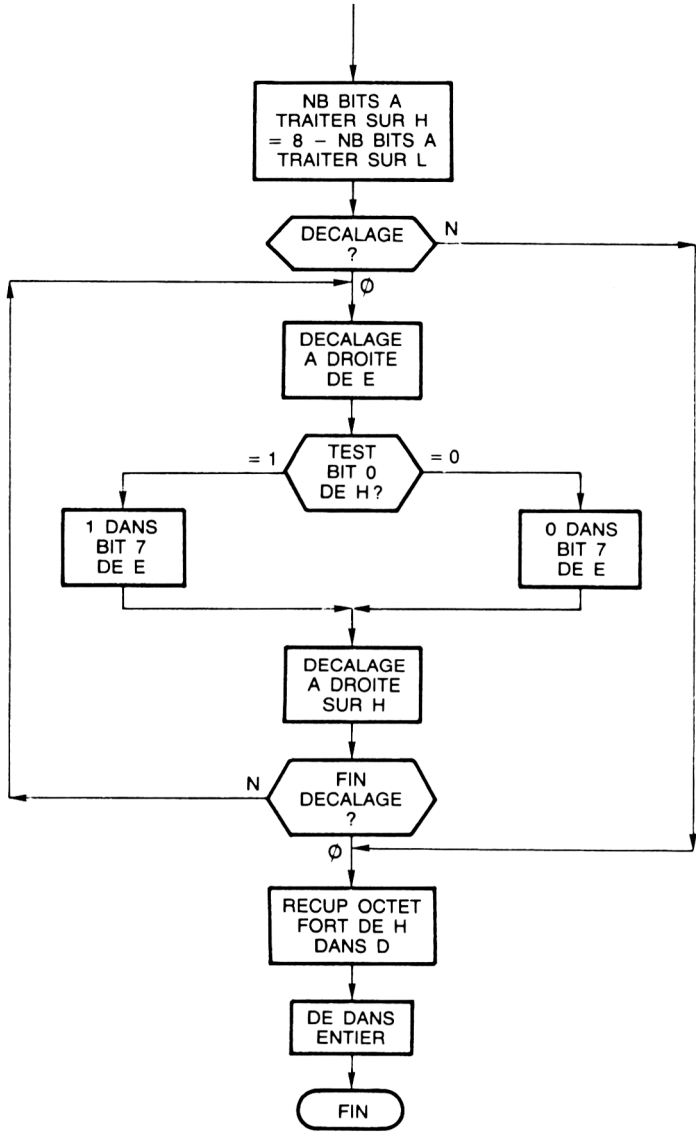
Module de traitement de récupération de la partie entière d'un flottant, dans le cas d'un exposant > 8 et < 16

Légende :

0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → lettre (pour « NON »)



(Début traitement octet 3)



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                 10 ;=* RECUP PARTIE ENTIERE D'UN FLOTTANT =*
                 20 ;=* SUR 16 BITS (SANS APPEL DE VECTEUR =*
                 30 ;=* MATHEMATIQUES)   /   TOUS CPC      =*
                 40 ;
7000             50         ent  #7000
7000             60         org  #7000
                 70 ;--* ROUT AFFICH POUR CPC 6128 *--
C37D             80 affich: equ  #c37d
                 90 ;--* CONNECT ROM HAUTE *--
B900            100 romhau: equ  #b900
                110 ;
7000  CD00B9    120         call romhau
                130 ;--* CHARG FLOT DANS ZFLOT *--
7003  DD219A70  140         ld   ix,zflot
7007  3EDD      150         ld   a,#dd
7009  DD7700    160         ld   (ix+0),a
700C  3E24      170         ld   a,#24
700E  DD7701    180         ld   (ix+1),a
7011  3EE7      190         ld   a,#e7
7013  DD7702    200         ld   (ix+2),a
7016  3E40      210         ld   a,#40
7018  DD7703    220         ld   (ix+3),a
701B  3E8F      230         ld   a,#8f
701D  DD7704    240         ld   (ix+4),a
                250 ;--* -#80 SUR OCTET EXPOSANT *--
7020  CBBF      260         res  7,a
                270 ;--* SAUVE EXPOSANT *--
7022  F5        280         push af
7023  DE08      290         sbc  a,8
7025  3016      300         jr   nc,gt8
                310 ;--* TRAIT SI EXP INF A 9 *--
7027  F1        320         pop  af
7028  2F        330         cpl
7029  3C        340         inc  a
                350 ;--* A = DECAL SUR OCTET 3 *--
702A  C608      360         add  a,8
                370 ;--* OCTET 3 DANS C *--
702C  DD4E03    380         ld   c,(ix+3)
                390 ;--* CORRECT BIT 7 OCTET 3 *--
702F  CBF9      400         set  7,c
                410 ;--* DECAL DROIT OCT 3 DANS C *--
7031  CB39      420 decall: srl  c
7033  3D        430         dec  a
7034  20FB      440         jr   nz,decall
```

```
7036  0600       450          ld   b,0
                 460 ;--* OBT ENTIER 16 BITS *--
7038  ED439F70   470          ld   (entier),bc
703C  C9         480          ret
                 490 ;
703D  F1         500 gt8:     pop  af
703E  F5         510          push af
703F  DE10       520          sbc  a,16
7041  3808       530          jr   c,gl16
                 540 ;--* EXP PLUS GRAND QUE 15 *--
7043  F1         550          pop  af
7044  21A170     560          ld   hl,overfl
7047  CD7DC3     570          call affich
704A  C9         580          ret
                 590 ;--* EXP ENTRE 9 ET 16 *--
704B  F1         600 gl16:    pop  af
704C  2F         610          cpl
704D  3C         620          inc  a
                 630 ;--* A = DECAL SUR OCTET 2 *--
704E  C610       640          add  a,16
7050  F5         650          push af
7051  DD219A70   660          ld   ix,zflot
                 670 ;--* RECUP OCTETS 2 ET 3 DANS L ET H *--
7055  DD6E02     680          ld   l,(ix+2)
7058  DD6603     690          ld   h,(ix+3)
                 700 ;--* CORR BIT 7 OCTET 3 DANS H *--
705B  CBFC       710          set  7,h
705D  110000     720          ld   de,0
7060  CB3D       730 decal2:  srl  l
7062  3D         740          dec  a
7063  20FB       750          jr   nz,decal2
7065  F1         760          pop  af
7066  2F         770          cpl
7067  3C         780          inc  a
                 790 ;--* NB BITS A TRAIT SUR H *--
7068  C608       800          add  a,8
706A  F5         810          push af
                 820 ;--* TRAIT OCTET 2 DE ZFLOT *--
706B  CB3B       830 testbl:  srl  e
706D  CB45       840          bit  0,l
706F  2804       850          jr   z,digitl
7071  CBFB       860          set  7,e
7073  1802       870          jr   finbl
7075  CBBB       880 digitl:  res  7,e
7077  CB3D       890 finbl:   srl  l
7079  3D         900          dec  a
707A  20EF       910          jr   nz,testbl
                 920 ;--* FIN TRAIT OCTET 2 *--
707C  F1         930          pop  af
```

```
707D  2F        940          cpl
707E  3C        950          inc  a
                960 ;--* NB BITS A TRAIT SUR DEB OCT 3 *--
707F  C608      970          add  a,8
7081  2811      980          jr   z,obtent
                990 ;--* TRAIT OCTET 3 DE ZFLOT *--
7083  CB3B     1000 testbh: srl  e
7085  CB44     1010         bit  0,h
7087  2804     1020         jr   z,digith
7089  CBFB     1030         set  7,e
708B  1802     1040         jr   finbh
708D  CBBB     1050 digith: res  7,e
708F  CB3C     1060 finbh:  srl  h
7091  3D       1070         dec  a
7092  20EF     1080         jr   nz,testbh
               1090 ;--* FIN OBT PART ENTIERE DE ZFLOT *--
7094  54       1100 obtent: ld   d,h
7095  ED539F70 1110         ld   (entier),de
7099  C9       1120         ret
               1130 ;
               1140 ;--* ZONES UTILISEES *--
709A           1150 zflot:  defs 5                    ;VALEUR FLOT
709F           1160 entier: defs 2                    ;RECUP PART ENT
70A1  6F766572 1170 overfl: defm "overflow"
70A9           1180 fintxt: defs 1
```

Pass 2 errors: 00

```
affich C37D  decal1 7031  decal2 7060
digith 708D  digit1 7075  entier 709F
finbh  708F  finbl  7077  fintxt 70A9
gl16   704B  gt8    703D  obtent 7094
overfl 70A1  romhau B900  testbh 7083
testbl 706B  zflot  709A
```

Table used: 226 from 900
Executes: 28672

ROUTINE DE RÉCUPÉRATION DE LA PARTIE DÉCIMALE D'UN FLOTTANT

Nous avons vu, dans la partie « lecture de la partie décimale », que pour obtenir 3 chiffres décimaux, avec une approximation au 9/1000e, le traitement des 6 premiers bits de la partie décimale du flottant suffit amplement.

Reprenons notre exemple de départ, soit : 12345.786

Rappel de la configuration hexadécimale :	DD	24	E7	40	8E
	0	1	2	3	4

Dans le cas général, et suivant la valeur de l'exposant, il sera traité ou les octets 3 et 2, ou les octets 2 et 1.

— cas d'un exposant < 8 : traitement des octets 3 et 2
— cas d'un exposant > 7 et < 16 : traitement des octets 2 et 1

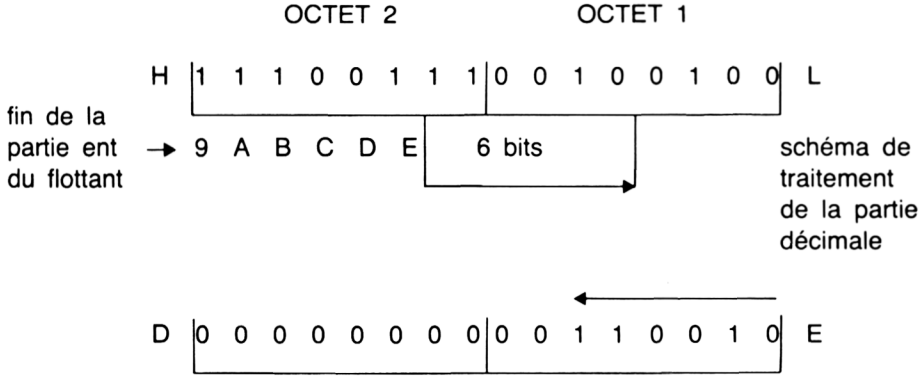
Dans notre exemple, l'exposant étant supérieur à 8, nous ferons un traitement sur les octets 2 et 1.
Ces 2 octets seront mis dans un double registre, soit dans le registre double HL en ce qui concerne la routine présentée.

La méthode de traitement de la partie décimale d'un flottant est à peu près similaire au traitement effectué pour la partie entière d'un flottant.

Nous allons procéder en 3 étapes successives :

— 1re étape : récupération des 6 bits dans le double registre DE
— 2e étape : multiplication du résultat par 10, puis ajout de la valeur 500 pour améliorer l'approximation.
— 3e étape : division du résultat par 2 à la puissance 6 pour obtenir la partie décimale du flottant.

Première étape :



Nous voyons que, comme pour le traitement de la partie entière, les 6 bits à récupérer se trouvent à cheval sur les octets H/L.

Le sens de la flèche de traitement sur HL étant différent du sens de traitement de la partie entière, nous testerons cette fois-ci chaque bit en bit 7, respectivement sur H, puis sur L ; puis nous positionnerons un bit 0 ou 1 dans le bit 0 de E.

Il nous faut donc positionner le premier des 2 bits sur H, en partant du bit 7 de cet octet. Ceci se fera par quelques décalages à gauche sur l'octet H.

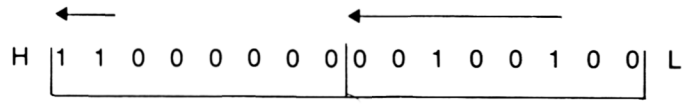
Après chaque bit traité, un décalage à gauche sur le registre H sera calculé de la manière suivante :

Le nombre de décalages à gauche à effectuer sur le registre H sera calculé de la manière suivante :

— si exposant − #80 < 8 : nb de décalages = exposant − #80
— si exposant − #80 > 7 et < 16 : décalage = (exposant − #80) − 8

Remarque : le nombre de décalages doit toujours être < 8

Dans notre exemple, le nombre de décalages sera égal à 6. Après décalage de 6 bits vers la gauche, sur le registre H, nous obtenons la configuration suivant de HL :



Avant d'effectuer le transfert dans DE, il est important de connaître le nombre de bits à traiter sur l'octet H.

Nombre de bits à traiter = 8 − nombre de décalages à gauche.

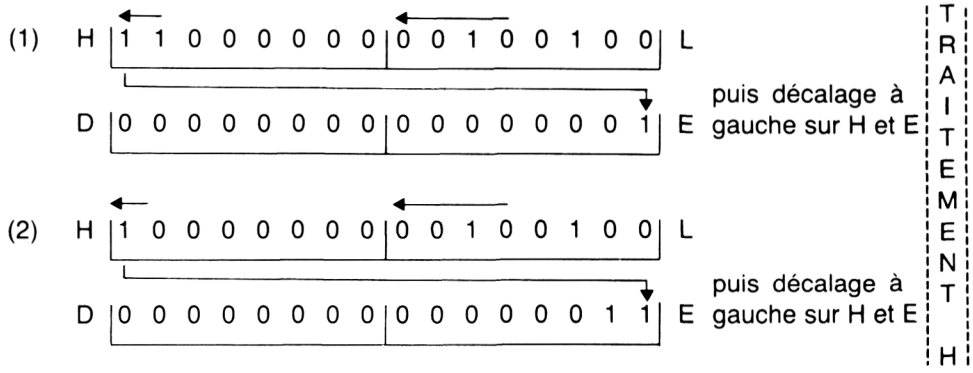
Cependant 2 cas peuvent se présenter :

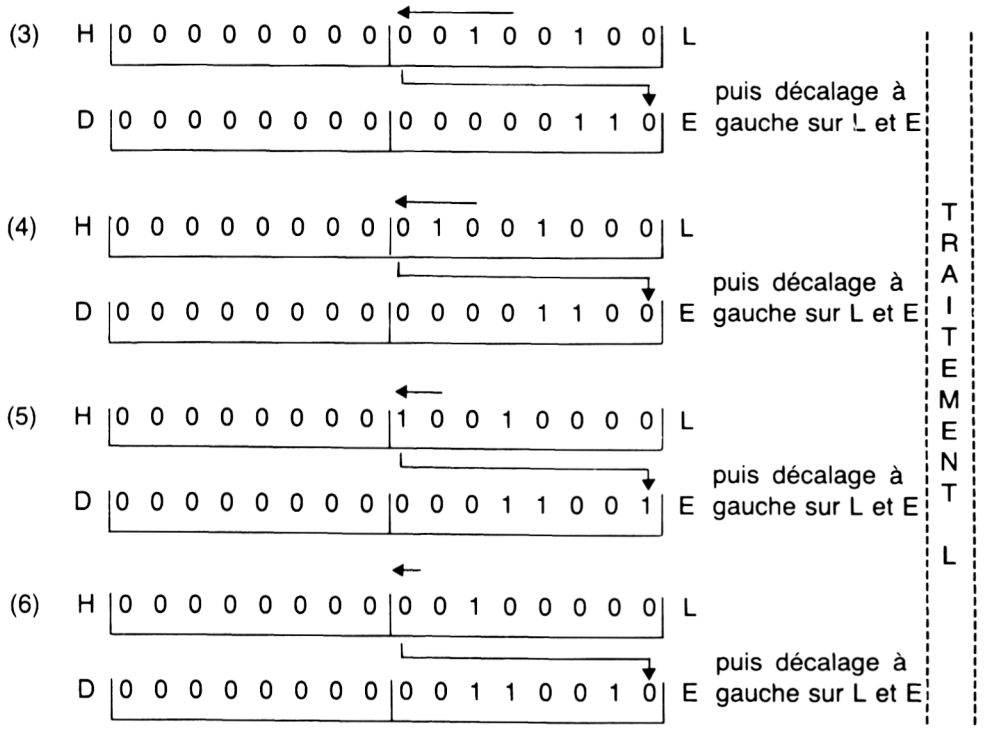
— le nb de bits à traiter est > 5 => ne traiter que l'octet H
— le nb de bits à traiter est < 6 => traiter les octets H et L.

Dans notre exemple, le nombre de bits à traiter sera égal à 2. Nous traiterons donc successivement les registres H puis L.

Schématiquement, nous aurons 6 étapes successives et similaires : les 2 premières étapes effectuent des décalages à gauche sur H ; les 4 dernières étapes effectuent des décalages à gauche sur L.

Schéma de traitement des 6 étapes



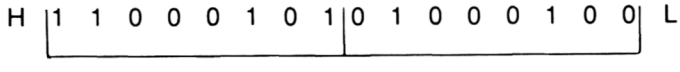


Deuxième étape :

Multiplication du résultat précédent par 10 à la puissance (nb décimales). En ce qui concerne la routine présentée, nous avons opté pour 3 décimales, ce qui implique que le contenu du registre DE sera multiplié par 1000.

Cette méthode permet d'obtenir un résultat < 65536, soit sur 2 octets ; toutefois nous ajouterons 500 au double registre DE, pour améliorer la précision du résultat final.

Après multiplication par 1000, nous obtenons #C350 dans DE, puis, après ajout de 500, nous obtenons #C544 dans HL, soit :

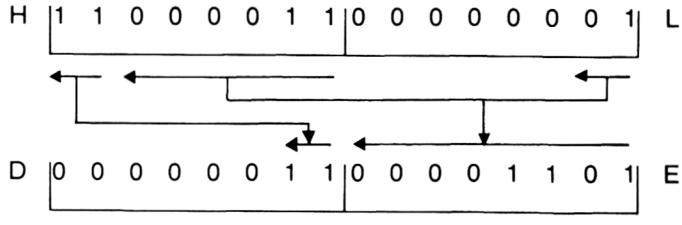


Troisième étape :

Division du résultat précédent par 2 à la puissance (nb de bits traités). Dans le cadre de notre méthode, nous avons choisi de traiter les 6 premiers bits de la partie décimale, ce qui implique que nous diviserons le registre HL par 64.

Or, une division par une puissance de 2 n'est autre qu'une suite de décalages à droite ; nous effectuerons donc 6 décalages sur HL. Mais voilà, nous ne savons faire des décalages que sur un seul registre à la fois...

Commençons donc par effectuer les 6 décalages à droite sur L, puis voyons le schéma de la suite du traitement.



Nous obtenons #30D dans DE, soit 789 en décimal, soit encore la valeur attendue pour la partie décimale de notre nombre flottant.

Nous voyons que la méthode à employer est identique à celle pour le traitement de la récupération de la partie entière (2e étape). Pour cela, il nous suffit simplement de fournir un décalage à faire sur L égal à 6, puis d'exécuter la partie de la routine de traitement de la partie entière, que nous avons intitulé.

Tronc commun partie entière/partie décimale (voir organigramme)

L'organigramme présenté ci-après ne retrace pas cette partie, mais y fait appel par un pavé « module ».

Pour s'y reporter, voir l'organigramme présenté pour la phase de traitement de la partie entière.

Conclusion sur la méthode employée

Cette méthode permet de mieux comprendre le mécanisme qui permet de récupérer les parties entière et décimale.

Il convient de préciser que d'autres méthodes de traitement existent, mais qui ne changent pas la philosophie du mécanisme.

A vos moments perdus, et afin de mettre en pratique ce qui vient d'être dit tout au long de cette section, je vous propose une autre méthode qui pourrait être un aboutissement de celle que nous venons de vous présenter.

Au lieu d'utiliser 2 doubles registres, il est possible, avec l'aide du report, de n'en utiliser qu'un seul.

Prenons le cas de la récupération de la partie entière pour étayer les explications qui vont suivre.

Au lieu de tester chaque bit en bit 0 de H, et afin d'éviter de décaler le registre L — disons sans aucune contrepartie — nous pourrions effectuer à la place le traitement suivant :

(1) décalage à droite de H à travers le report,
(2) rotation à droite de L à travers le report,
(3) répéter les opérations (1) et (2) autant de fois que l'indique le nombre de décalages calculé.

Le gain de cette méthode est évident, puisqu'il permet de ne traiter qu'un seul registre double, et d'économiser la phase de test de chaque bit...

Calcul de l'erreur relative obtenue avec la méthode présentée :

Nous avons effectué quelques essais sur le flottant traité en exemple, afin de voir quelle pouvait être l'erreur relative absolue (en 1/1 000e) sur l'obtention de 3 décimales ; avec un choix de traitement des 6 premiers bits de la partie décimale.

Pour ce faire, nous ne ferons varier que l'exposant du flottant.

Soit le tableau de résultat obtenu après essais :

Exposant	Partie décimale (3 chiffres)	Partie décimale (avec routine)	erreur relative absolue
#80	753	757	4
#81	507	507	0
#82	014	007	7
#83	028	023	5
#84	056	054	2
#85	112	117	5
#86	225	226	1
#87	451	445	6
#88	902	898	4
#89	805	804	1
#8A	611	617	6
#8B	223	226	3
#8C	446	445	1
#8D	893	898	5
#8E	786	789	3
#8F	572	570	2

On constatera que l'erreur relative absolue est en moyenne comprise entre 3 et 4 millièmes, ce qui reste acceptable.

Bien entendu, il est tout à fait possible de traiter 16 bits au lieu de 6, mais le choix de la méthode présentée ne le permet pas facilement ; aussi, avec la deuxième proposition, nous devrions y arriver plus aisément.
Il n'est cependant pas certain que pour un choix de 3 décimales, le traitement de 16 bits nous donne de meilleurs résultats.

Limites de la routine présentée :

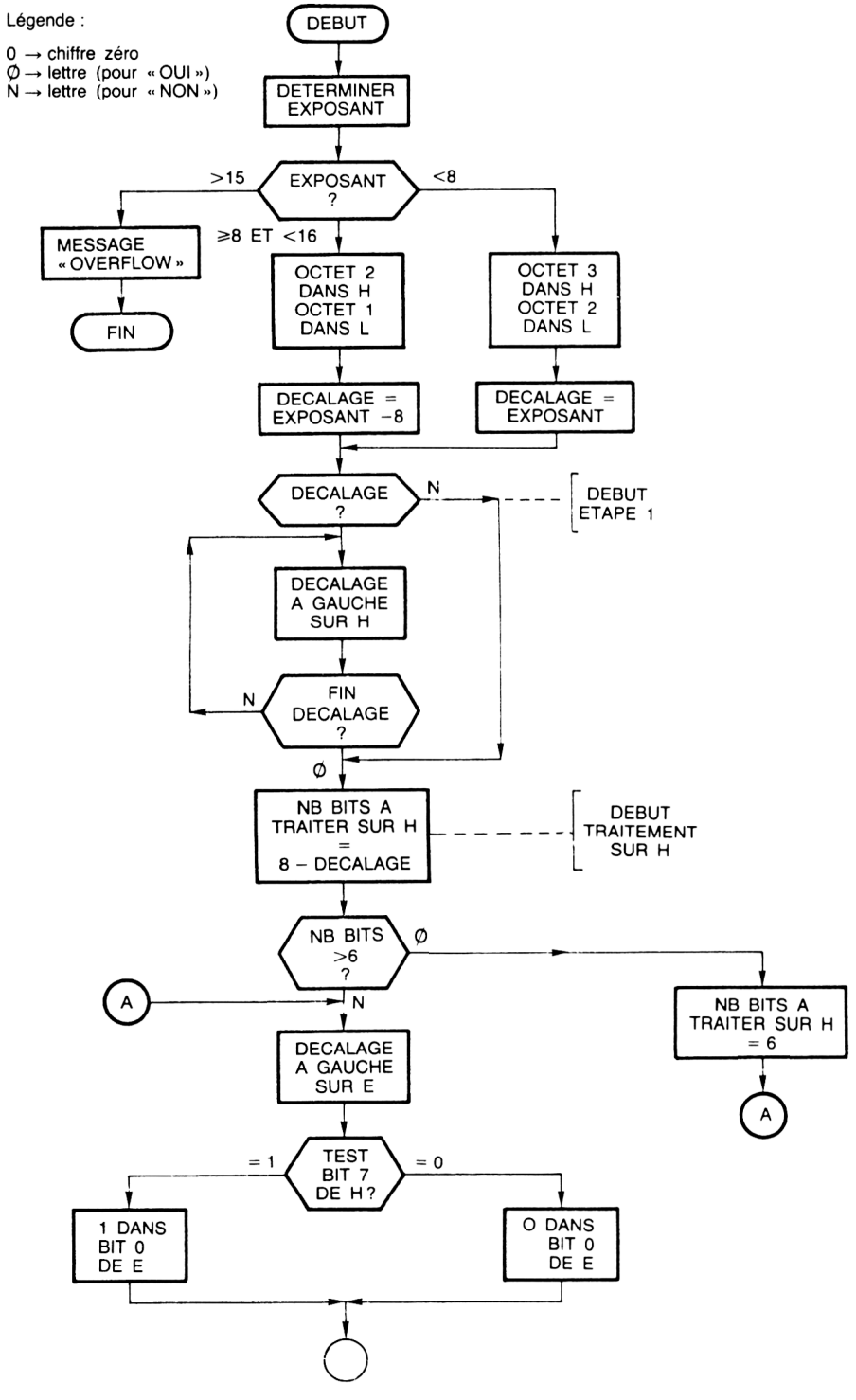
La routine présentée ci-après ne permet de traiter que des valeurs décimales supérieures ou égales à 0.5.
En effet, pour les valeurs entre 0 et 0.5 non incluses, l'exposant devient inférieur à #80.
Mais vous pouvez améliorer la routine pour pallier cette lacune, puisque vous avez tous les éléments nécessaires...

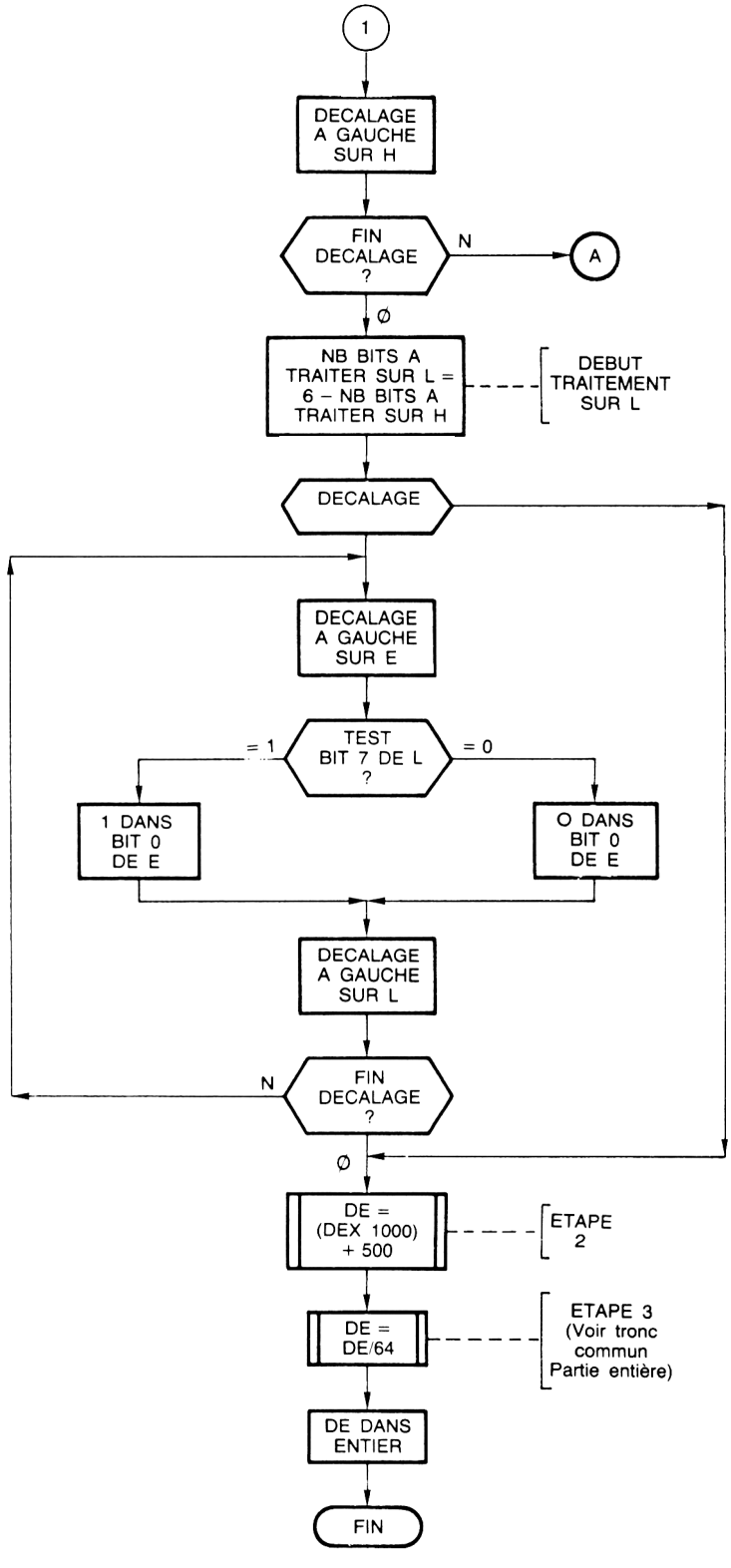
Par ailleurs, seuls les chiffres significatifs sont retenus pour la récupération de la partie décimale.
En effet, les exemples suivants reflètent cet état de fait, sur l'exemple traité :

— pour un exposant de	#82	nous obtenons	7	au lieu de	007
—	#83	.....	23	.....	023
—	#84	.....	54	.....	054

Mais puisque nous savons, par hypothèse, que nous devons obtenir 3 chiffres décimaux, il nous suffit de rajouter le nombre voulu de 0 non significatifs dans le buffer, mais qui sont en fait significatifs en regard du nombre réel traité.

Routine de récupération de la partie décimale d'un flottant




1
DECALAGE A GAUCHE SUR H
FIN DECALAGE ?
N
A
Ø
NB BITS A TRAITER SUR L = 6 – NB BITS A TRAITER SUR H
DEBUT TRAITEMENT SUR L
DECALAGE
DECALAGE A GAUCHE SUR E
TEST BIT 7 DE L ?
= 1
= 0
1 DANS BIT 0 DE E
O DANS BIT 0 DE E
DECALAGE A GAUCHE SUR L
FIN DECALAGE ?
N
Ø
DE = (DEX 1000) + 500
ETAPE 2
DE = DE/64
ETAPE 3 (Voir tronc commun Partie entière)
DE DANS ENTIER
FIN

Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                 10 ;=* RECUP PARTIE DECIMALE D'UN FLOTTANT =*
                 20 ;=* SUR 16 BITS (SANS APPEL DE VECTEUR =*
                 30 ;=* MATHEMATIQUES)   /   TOUS CPC      =*
                 40 ;
7000             50         ent  #7000
7000             60         org  #7000
                 70 ;--* ROUT AFFICH POUR CPC 6128 *--
C37D             80 affich: equ  #c37d
                 90 ;--* CONNECT ROM HAUTE *--
B900            100 romhau: equ  #b900
                110 ;
7000  CD00B9    120         call romhau
                130 ;--* CHARG FLOT DANS ZFLOT *--
7003  DD21EA70  140         ld   ix,zflot
7007  3EDD      150         ld   a,#dd
7009  DD7700    160         ld   (ix+0),a
700C  3E24      170         ld   a,#24
700E  DD7701    180         ld   (ix+1),a
7011  3EE7      190         ld   a,#e7
7013  DD7702    200         ld   (ix+2),a
7016  3E40      210         ld   a,#40
7018  DD7703    220         ld   (ix+3),a
701B  3E89      230         ld   a,#89
701D  DD7704    240         ld   (ix+4),a
7020  CBBF      250         res  7,a
7022  4F        260         ld   c,a
7023  F5        270         push af
7024  DE09      280         sbc  a,9
7026  300C      290         jr   nc,gt8
                300 ;--* TRAIT SI EXP LT 8 *--
7028  DD7E02    310         ld   a,(ix+2)
702B  6F        320         ld   l,a
702C  DD7E03    330         ld   a,(ix+3)
702F  67        340         ld   h,a
7030  CBFC      350         set  7,h
7032  181B      360         jr   decal
7034  F1        370 gt8:    pop  af
7035  F5        380         push af
7036  DE10      390         sbc  a,16
7038  3808      400         jr   c,lt16
                410 ;--* TRAIT SI EXP GT 15 *--
703A  F1        420         pop  af
703B  21F170    430         ld   hl,overfl
703E  CD7DC3    440         call affich
7041  C9        450         ret
                460 ;--* TRAIT SI EXP ENTRE 8 ET 15 *--
```

```
7042  F1        470 lt16:   pop  af
7043  F5        480         push af
7044  DE08      490         sbc  a,8
7046  4F        500         ld   c,a
7047  DD7E01    510         ld   a,(ix+1)
704A  6F        520         ld   l,a
704B  DD7E02    530         ld   a,(ix+2)
704E  67        540         ld   h,a
                550 ;--* TRAIT ETAPE 1 *--
704F  F1        560 decal:  pop  af
7050  79        570         ld   a,c
7051  F5        580         push af
7052  DE00      590         sbc  a,0
7054  2805      600         jr   z,fdl
                610 ;--* DECAL A GAUGHE SUR H *--
7056  CB24      620 fdecl:  sla  h
7058  3D        630         dec  a
7059  20FB      640         jr   nz,fdecl
705B  F1        650 fdl:    pop  af
705C  2F        660         cpl
705D  3C        670         inc  a
705E  C608      680         add  a,8
7060  F5        690         push af
                700 ;--* NB BITS A TRAITER SUR H *--
7061  DE06      710         sbc  a,6
7063  CB7F      720         bit  7,a
7065  2848      730         jr   z,six
7067  F1        740         pop  af
7068  F5        750 corra:  push af
                760 ;--* TRAIT A BITS SUR H *-
7069  110000    770         ld   de,0
706C  CB23      780 fdech:  sla  e
706E  CB7C      790         bit  7,h
7070  2804      800         jr   z,digh
7072  CBC3      810         set  0,e
7074  1802      820         jr   fbh
7076  CB83      830 digh:   res  0,e
7078  CB24      840 fbh:    sla  h
707A  3D        850         dec  a
707B  20EF      860         jr   nz,fdech
707D  F1        870         pop  af
707E  2F        880         cpl
707F  3C        890         inc  a
7080  C606      900         add  a,6
7082  2815      910         jr   z,fbl
                920 ;--* PAS DE TRAIT SUR L SI A NEG *--
7084  CB7F      930         bit  7,a
7086  2011      940         jr   nz,fbl
                950 ;--* TRAIT A BITS SUR L *--
```

```
7088  CB23      960 fdecl:  sla  e
708A  CB7D      970         bit  7,l
708C  2804      980         jr   z,digl
708E  CBC3      990         set  0,e
7090  1802     1000         jr   finl
7092  CB83     1010 digl:   res  0,e
7094  CB25     1020 finl:   sla  l
7096  3D       1030         dec  a
7097  20EF     1040         jr   nz,fdecl
               1050 ;--* TRAIT ETAPE 2 *--
7099  3E03     1060 fbl:    ld   a,3
709B  62       1070         ld   h,d
709C  6B       1080         ld   l,e
709D  19       1090 multl0: add  hl,de
709E  29       1100         add  hl,hl
709F  19       1110         add  hl,de
70A0  29       1120         add  hl,hl
70A1  54       1130         ld   d,h
70A2  5D       1140         ld   e,l
70A3  3D       1150         dec  a
70A4  20F7     1160         jr   nz,multl0
               1170 ;--* +500 DANS HL *--
70A6  11FA00   1180         ld   de,250
70A9  19       1190         add  hl,de
70AA  19       1200         add  hl,de
               1210 ;--* TRAIT ETAPE 3 *--
70AB  CDB470   1220         call etap3
70AE  C9       1230         ret
               1240 ;--* CORR A SI SUP A 6 *--
70AF  F1       1250 six:    pop  af
70B0  3E06     1260         ld   a,6
70B2  18B4     1270         jr   corra
               1280 ;
               1290 ;--* DE = DE / 64 (ETAPE 3) *--
               1300 ;--* 6 DECAL A DROITE SUR H *--
70B4  3E06     1310 etap3:  ld   a,6
70B6  110000   1320         ld   de,0
70B9  CB3D     1330 decal2: srl  l
70BB  3D       1340         dec  a
70BC  20FB     1350         jr   nz,decal2
               1360 ;--* TRAIT OCTET 2 DE ZFLOT *--
70BE  3E02     1370         ld   a,2
70C0  CB3B     1380 testbl: srl  e
70C2  CB45     1390         bit  0,l
70C4  2804     1400         jr   z,digitl
70C6  CBFB     1410         set  7,e
70C8  1802     1420         jr   finbl
70CA  CBBB     1430 digitl: res  7,e
70CC  CB3D     1440 finbl:  srl  l
```

```
70CE  3D        1450          dec  a
70CF  20EF      1460          jr   nz,testbl
                1470 ;--* TRAIT 6 BITS SUR H *--
70D1  3E06      1480          ld   a,6
70D3  CB3B      1490 testbh:  srl  e
70D5  CB44      1500          bit  0,h
70D7  2804      1510          jr   z,digith
70D9  CBFB      1520          set  7,e
70DB  1802      1530          jr   finbh
70DD  CBBB      1540 digith:  res  7,e
70DF  CB3C      1550 finbh:   srl  h
70E1  3D        1560          dec  a
70E2  20EF      1570          jr   nz,testbh
                1580 ;--* FIN OBT PART DEC DE ZFLOT *--
70E4  54        1590 obtent:  ld   d,h
70E5  ED53EF70  1600          ld   (entier),de
70E9  C9        1610          ret
                1620 ;
                1630 ;--* ZONES UTILISEES *--
70EA            1640 zflot:   defs 5                    ;VALEUR FLOT
70EF            1650 entier:  defs 2                    ;RECUP PART DEC
70F1  6F766572  1660 overfl:  defm "overflow"
70F9            1670 fintxt:  defs 1


Pass 2 errors: 00

affich C37D  corra  7068  decal  704F
decal2 70B9  digh   7076  digith 70DD
digitl 70CA  digl   7092  entier 70EF
etap3  70B4  fbh    7078  fbl    7099
fdl    705B  fdecl  7056  fdech  706C
fdecl  7088  finbh  70DF  finbl  70CC
finl   7094  fintxt 70F9  gt8    7034
lt16   7042  multl0 709D  obtent 70E4
overfl 70F1  romhau B900  six    70AF
testbh 70D3  testbl 70C0  zflot  70EA


Table used:   371  from   900
Executes: 28672
```

CHAPITRE 3

TRAITEMENT DES LISTES

PRÉLIMINAIRE

Il nous a semblé intéressant de consacrer un chapitre à un sujet peu abordé jusqu'à présent pour l'Amstrad CPC, et qui pourtant offre des possibilités d'application fort utiles.

En effet le traitement des listes fait partie intégrante de notre vie de tous les jours ; que ce soit en gestion familiale ou en entreprise, ou encore dans des opérations de classement de fiches de toute nature, ou bien encore dans le domaine du traitement de texte (mailing etc.).

Certes, nous ne prétendons pas épuiser le sujet, notre but étant avant tout de présenter des méthodes vous permettant d'aller plus loin dans le sujet ; celles-ci vous donnant une approche technique non négligeable de ce qui peut se faire dans ce domaine très utilisé en micro-informatique.

Programmation des routines

Les routines présentées sont directement utilisables dans des programmes Basic de votre cru, par simple appel d'une instruction de type RSX, avec passage des paramètres demandés.

Bien évidemment, ces routines s'intégreront parfaitement dans vos programmes assembleur, et pourront être appelées aisément par une instruction CALL (voir présentation de l'ouvrage).

Routines traitées dans ce chapitre

(1) recherche d'un texte dans une zone mémoire donnée (FIND)
(2) changement d'un texte par un autre texte (CHANGE)
(3) tri de tableau de chaînes, en DIM ou de zones particulières
(4) ..... réels, ...
(5) ..... d'entiers, ...

RECHERCHE D'UNE CHAÎNE DANS UNE ZONE DONNÉE

La routine présentée ci-après permet de rechercher un texte d'une longueur maximale de 255 caractères, dans une portion de mémoire que vous lui aurez indiqué.

Le texte à chercher peut être un mot, un groupe de mots, ou tout simplement une suite de caractères quelconques.

Toutefois, une entrée à vide ou une chaîne constituée d'espaces ne peut être entrée ; dans ce cas, la longueur du texte à chercher devient nulle, et un message d'erreur, apparaît à l'écran.

Pour effectuer la recherche, vous devez indiquer l'adresse d'un texte à chercher ; ce texte doit obligatoirement se terminer par un 0 binaire, puisque c'est par ce moyen que la routine détecte la fin du texte.

Il vous faudra également indiquer à la routine les adresses de début et de fin de la zone ou devra s'effectuer la recherche. Pour ce faire, vous devrez fournir l'instruction RSX suivante :

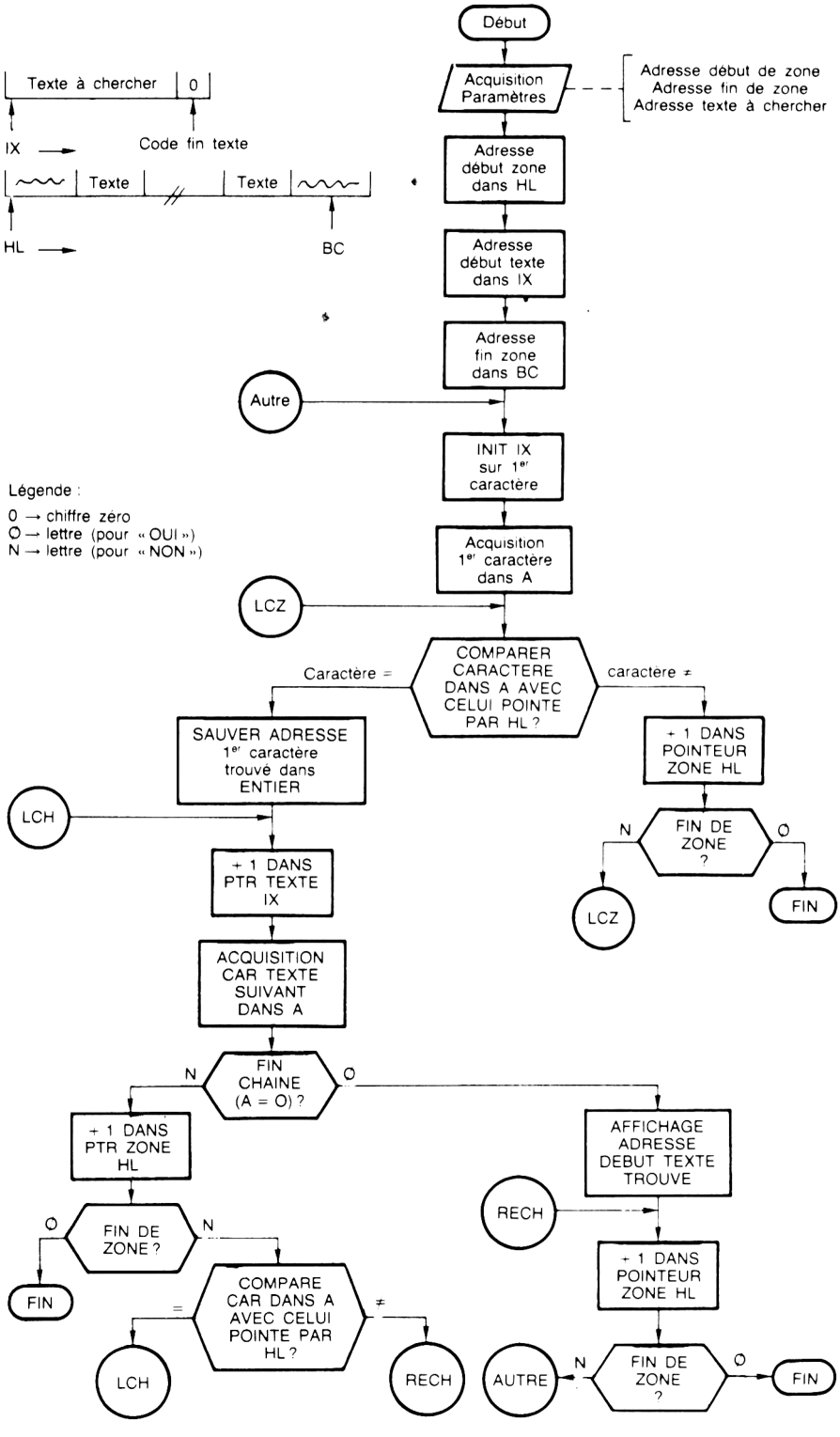
```
| FIND,adr début zone, adr fin zone,adr texte à chercher
```

Le principe de traitement de la routine est fort simple. La routine pointe sur le premier caractère du texte à chercher, puis progresse dans la zone donnée jusqu'à ce qu'elle trouve ce caractère ; puis la routine pointe sur le caractère suivant du texte à chercher, et regarde si celui-ci suit le précédent dans la zone donnée ; si c'est le cas la routine recommence ce même processus jusqu'à obtention du texte cherché, si celui-ci est en effet présent à cet endroit de la zone.

Si ce n'est pas le cas, la routine se débranche pour se repointer sur le premier caractère du texte à chercher ; puis elle continue de progresser dans la zone donnée, jusqu'à ce qu'elle trouve à nouveau ce premier caractère, et recommence le processus. Bien entendu, si la routine détecte la fin de la zone donnée, le traitement est terminé.

A chaque fois que la routine détecte le premier caractère du texte dans la zone donnée, elle mémorise l'adresse de celui-ci, elle pourra alors afficher celle-ci ultérieurement, dans le cas ou cette recherche serait fructueuse ; si plusieurs textes existent dans la zone donnée, la routine les détectera tous.

Routine de recherche d'un texte donné dans une liste



```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                 10 ;=* ROUTINE FIND POUR TOUS CPC *=
6FA0             20         ent  #6fa0
6FA0             30         org  #6fa0
                 40 ;--* TRAITEMENT RSX *--
BCD1             50 rsx:    equ  #bcd1
6FA0  01AA6F     60         ld   bc,ext
6FA3  21C86F     70         ld   hl,space
6FA6  CDD1BC     80         call rsx
6FA9  C9         90         ret
6FAA  C36F      100 ext:    defw nom
                110 ;--* RECUPERATION PARAMETRES *--
                120 ;--* HL=DEBZ / BC=FINZ / DE=DEBTXT *--
6FAC  DD6605    130         ld   h,(ix+5)
6FAF  DD6E04    140         ld   l,(ix+4)
6FB2  DD4603    150         ld   b,(ix+3)
6FB5  DD4E02    160         ld   c,(ix+2)
6FB8  DD5601    170         ld   d,(ix+1)
6FBB  DD5E00    180         ld   e,(ix+0)
6FBE  FE03      190         cp   3
6FC0  C0        200         ret  nz
6FC1  183D      210         jr   rout
                220 ;
6FC3  46494E    230 nom:    defm "FIN"
6FC6  C4        240         defb "D"+#80
6FC7  00        250         defb 0
6FC8            260 space:  defs 4
                270 ;
                280 ;--* DEBUT ROUTINE 'FIND' *--
7000            290         org  #7000
B900            300 romhau: equ  #b900
C37D            310 affich: equ  #c37d
7000  ED539870  320 rout:   ld   (adrtxt),de
7004  CD00B9    330         call romhau
                340 ;--* TEST SI LONG TEXTE NUL *--
7007  EB        350         ex   de,hl
7008  3E00      360         ld   a,0
700A  BE        370         cp   (hl)
700B  2822      380         jr   z,err
700D  EB        390         ex   de,hl
                400 ;--* ADR TXT DANS IX *--
700E  DD2A9870  410 autre:  ld   ix,(adrtxt)
                420 ;--* CAR TEXTE DANS A *--
7012  DD7E00    430         ld   a,(ix+0)
                440 ;--* TEST A AVEC CAR ZONE *--
7015  BE        450 lcz:    cp   (hl)
```

```
7016  202D      460          jr   nz,difl
                470 ;--* SAVE ADR 1ER CAR TROUVE *--
7018  229670    480          ld   (entier),hl
                490 ;--* CAR TXT SUIV DANS A *--
701B  DD23      500 lch:     inc  ix
701D  DD7E00    510          ld   a,(ix+0)
                520 ;--* TEST FIN TEXTE *--
7020  FE00      530          cp   0
7022  2812      540          jr   z,trouve
7024  CD4B70    550          call finzon
7027  2805      560          jr   z,ntrouv
7029  BE        570          cp   (hl)
702A  28EF      580          jr   z,lch
702C  1811      590          jr   rech
702E  C9        600 ntrouv: ret
                610 ;--* MESSAGE ERREUR SI LONG NUL *--
702F  21A170    620 err:     ld   hl,merr
7032  CD7DC3    630          call affich
7035  C9        640          ret
                650 ;--* AFF ADR TEXTE TROUVE *--
                660 ;--* SAVE REGISTRES *--
7036  E5        670 trouve: push hl
7037  C5        680          push bc
7038  D5        690          push de
7039  CD5670    700          call affasc
                710 ;--* RECUP REGISTRES *--
703C  D1        720          pop  de
703D  C1        730          pop  bc
703E  E1        740          pop  hl
703F  CD4B70    750 rech:    call finzon
7042  20CA      760          jr   nz,autre
7044  C9        770          ret
7045  CD4B70    780 difl:    call finzon
7048  20CB      790          jr   nz,lcz
704A  C9        800          ret
                810 ;--* +1 DANS PTR ZONE ET TEST SI FIN *--
                820 ;--* PTR ZONE DANS DE *--
704B  23        830 finzon: inc  hl
704C  E5        840          push hl
704D  D1        850          pop  de
                860 ;--* SAVE PTR ZONE *--
704E  E5        870          push hl
                880 ;--* FIN ZONE DANS HL *--
704F  C5        890          push bc
7050  E1        900          pop  hl
                910 ;--* CALCUL SI FIN DE ZONE *--
7051  A7        920          and  a
7052  ED52      930          sbc  hl,de
                940 ;--* RECUP PTR ZONE *--
```

```
7054  E1          950          pop  hl
7055  C9          960          ret
                  970 ;--* AFF DECIMALE ADR TEXTE TROUVE *--
7056  ED4B9670    980 affasc: ld   bc,(entier)
705A  DD219A70    990          ld   ix,chaine
705E  111027     1000          ld   de,10000
7061  CD8370     1010          call ascii
7064  11E803     1020          ld   de,1000
7067  CD8370     1030          call ascii
706A  116400     1040          ld   de,100
706D  CD8370     1050          call ascii
7070  110A00     1060          ld   de,10
7073  CD8370     1070          call ascii
7076  110100     1080          ld   de,1
7079  CD8370     1090          call ascii
707C  219A70     1100          ld   hl,chaine
707F  CD7DC3     1110          call affich
7082  C9         1120          ret
7083  3E2F       1130 ascii:   ld   a,#2f
7085  60         1140          ld   h,b
7086  69         1150          ld   l,c
7087  3C         1160 compte:  inc  a
7088  ED52       1170          sbc  hl,de
708A  30FB       1180          jr   nc,compte
708C  ED5A       1190          adc  hl,de
708E  44         1200          ld   b,h
708F  4D         1210          ld   c,l
7090  DD7700     1220          ld   (ix+0),a
7093  DD23       1230          inc  ix
7095  C9         1240          ret
                 1250 ;--* ZONES UTILISEES PAR LA ROUTINE *--
7096             1260 entier: defs 2
7098             1270 adrtxt: defs 2
                 1280 ;--* BUFFER ET MESSAGE *--
709A             1290 chaine: defs 5
709F  0A00       1300 fchain: defb 10,0                ;CODE LIGNE SUIV
70A1  74657874   1310 merr:   defm "texte entre incorrect / arret"
70BE  0A00       1320 fmerr:  defb 10,0                ;CODE LIGNE SUIV
```

```
Pass 2 errors: 00

adrtxt 7098  affasc 7056  affich C37D
ascii  7083  autre  700E  chaine 709A
compte 7087  difl   7045  entier 7096
err    702F  ext    6FAA  fchain 709F
finzon 704B  fmerr  70BE  lch    701B
lcz    7015  merr   70A1  nom    6FC3
ntrouv 702E  rech   703F  romhau B900
rout   7000  rsx    BCD1  space  6FC8
trouve 7036

Table used:   308  from   900
Executes: 28576
```

ROUTINE DE REMPLACEMENT D'UN TEXTE PAR UN AUTRE

La routine présentée ci-après permet d'effectuer le remplacement d'un texte d'une zone donnée, par un autre texte. Ce nouveau texte peut être soit plus petit, soit plus long que l'ancien texte de la zone donnée. De même que pour le "FIND", le texte ancien ou nouveau peut être un mot, un groupe de mots, ou tout simplement une suite de caractères quelconques.

La longueur du nouveau texte est théoriquement fixée à 255 ; mais il est fortement recommandé de faire en sorte que la différence de longueur entre nouveau texte et ancien texte ne dépasse pas les 30 caractères ; en effet, si le texte à changer est très fréquent dans la zone donnée, il pourrait en résulter un léger effaçage au-dessus de la zone donnée.

Nous verrons un peu plus loin la raison de ce petit problème.

Ici aussi, toute entrée vide ou suite d'espaces sera interprétée comme un texte ayant une longueur nulle, et provoquera de ce fait un message d'erreur suivi de l'arrêt du traitement.

Pour effectuer le remplacement d'un ou plusieurs textes identiques par un nouveau texte, il vous faudra passer un certain nombre de paramètres à la routine "CHANGE".

(1) adresse de début de la zone donnée
(2) adresse de fin de la zone donnée
(3) adresse du premier caractère du texte à chercher (ou ANCIEN)
(4) adresse du premier caractère du NOUVEAU texte

Pour ce faire, vous devrez fournir l'instruction RSX suivante :

```
| CHANGE,adr déb zone,adr fin zone,adr txt ancien,adr txt nouveau
```

La routine doit être capable de détecter les changements de texte qui sont intempestifs, dans le cas ou le texte nouveau est plus long que le texte ancien. Pour cela, il est impératif d'initialiser la zone donnée, avec des 0 binaires, avant réception des informations à traiter par la routine de remplacement de texte.

C'est grâce à ces 0 binaires que la routine sera alertée en cas de trop-plein de la zone donnée.

Pour ce faire, et avant changement de texte, la routine analyse les 30 derniers caractères de la zone donnée ; si ceux-ci sont toujours à 0, alors le changement de texte s'effectuera, sinon un message d'erreur sera provoqué, suivi de l'arrêt du traitement. En effet, dans le cas d'un nouveau texte plus long que l'ancien, la routine effectue un transfert du texte vers le haut de la zone donnée, afin de libérer la place nécessaire pour insérer le nouveau texte.

Nous pouvons donc, maintenant, imaginer facilement ce qui se passera si un nouveau texte a plus de 30 caractères que l'ancien, dans le cas d'un changement intempestif de texte.

Imaginons par exemple le changement d'un texte de 10 caractères par un texte de 200 caractères.

Si la portion de zone à 0 binaire est assez grande, tout va bien se passer durant les premiers changements ; puis au bout du Nième changement, nous allons approcher la zone limite des 30 derniers caractères ; au moment où le prochain changement devra se faire, la routine va analyser les 30 derniers caractères et décrétera qu'il n'y a pas de problème, puis elle effectuera le changement. Mais voilà, 190 caractères supplémentaires vont accroître la zone donnée, et bien évidemment détruire 190 octets au-dessus cette zone donnée.

Vous comprenez maintenant pourquoi il est fort recommandé de faire en sorte que le nouveau texte n'excède pas l'ancien texte de plus de 30 caractères.
Oui mais, me direz-vous, il suffit d'augmenter la « zone d'alerte » qui se trouve en bout de la zone donnée ; bien entendu, ceci est tout à fait possible, et vous pourriez par exemple la mettre au maximum, soit à 255 zéros binaire.
Ceci dépendra essentiellement de l'utilisation que vous ferez de cette routine.

Si cela vous tente, il suffit pour y parvenir de modifier l'instruction LD E, 30 en 1220, en remplaçant la valeur 30 par toute autre valeur inférieure ou égale à 255.

En ce qui concerne les textes ancien et nouveau, vous devez, tout comme la routine "FIND", mettre un 0 binaire en fin de texte.

Mais voyons maintenant comment fonctionne cette routine. Il est bien évident que la phase de recherche correspond au traitement effectué par la routine "FIND".
Mais une fois notre texte trouvé, 2 cas peuvent se présenter :

(1) la longueur du nouveau texte est plus petite que la longueur de l'ancien texte.

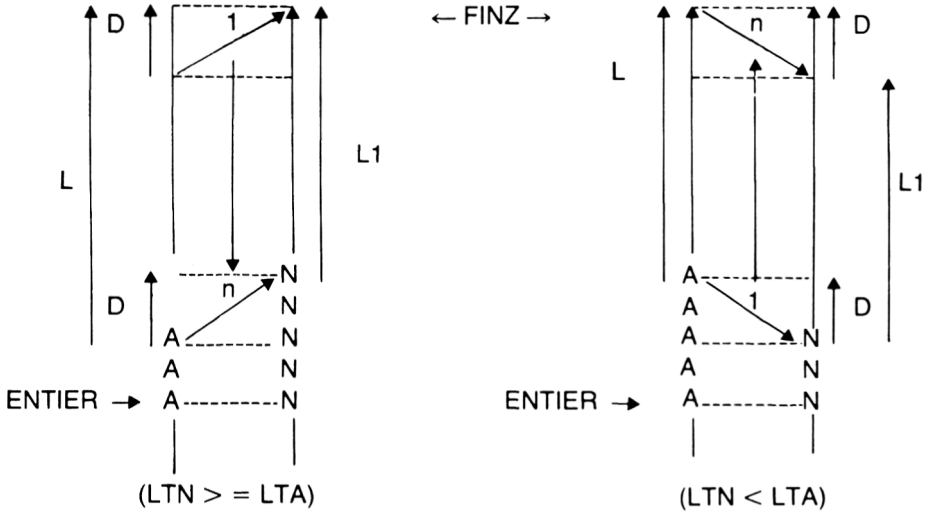
(2) la longueur du nouveau texte est plus grande que la longueur de l'ancien texte (ou égale).

Afin d'éviter de se perdre dans un long discours, et surtout pour se rapprocher le plus possible de la routine présentée, nous allons vous présenter un schéma de fonctionnement de la routine. Pour ce faire, un certain nombre d'hypothèses est indispensable.

Variables utilisées :

ENTIER : pointe sur le premier caractère de l'ancien texte
FINZ : adresse de fin de la zone donnée
LTA : longueur de l'ancien texte
LTN : longueur du nouveau texte
D : différence entre LTN et LTA (D = LTN – LTA)
L : longueur restante entre la fin de l'ancien texte et FINZ
L1 : longueur de la zone à transférer pour libérer la place pour le nouveau texte.

Schéma de fonctionnement :



La routine commence par effectuer un "FIND" du texte à changer, puis regarde si la zone d'alerte est à 0 binaire.

Si la zone d'alerte ne signale pas de "CHANGE" intempestifs, elle commence par calculer D = LTN – LTA.

Puis, que nous soyons dans le cas D négatif, ou D positif ou nul, il va falloir effectuer le transfert du reste de la zone, qui est au-dessus de l'ancien texte, après le dernier caractère du nouveau texte ; ceci se fera par une instruction de transfert de bloc. Mais le transfert se fera différemment selon que nous sommes dans l'un ou l'autre cas.

Si D est négatif, nous sommes dans le cas ou la longueur du texte nouveau est plus petite que celle du texte ancien.
Dans ce cas (voir schéma de fonctionnement), le transfert se fera de la manière suivante :

SOURCE = ENTIER + LTA
DESTINATION = ENTIER + LTN
COMPTEUR = FINZ – ENTIER – LTA

Puis nous devons effectuer le transfert de bas en haut, soit de 1 à n (voir schéma de fonctionnement) ; l'instruction assembleur utilisée sera donc un LDIR.

Si D est positif, nous sommes dans le cas ou la longueur du texte nouveau est plus grande que celle du texte ancien (ou égale).
Dans ce cas (voir schéma de fonctionnement), le transfert se fera de la manière suivante :

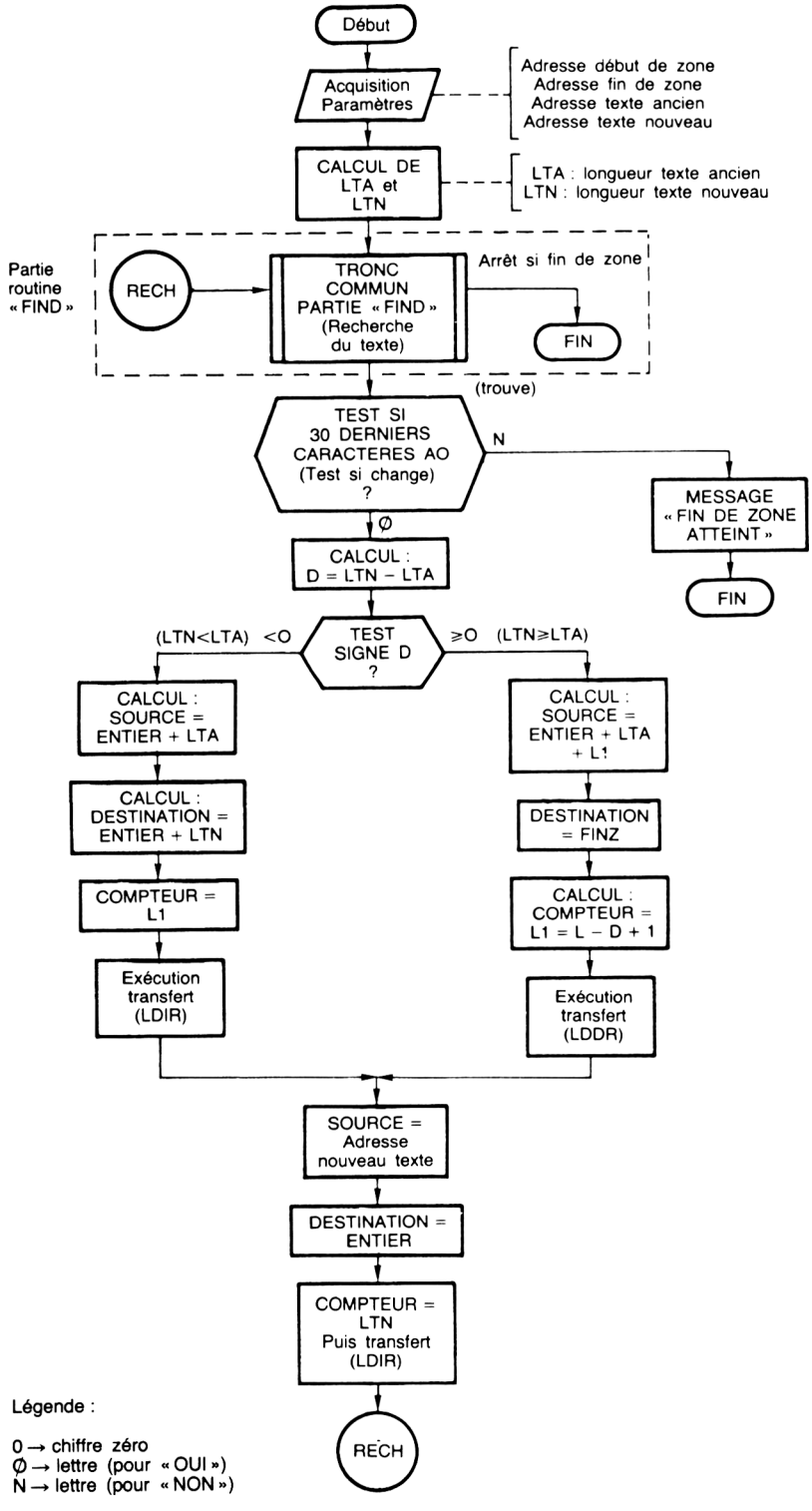
SOURCE = ENTIER + LTA + (L − D) = FINZ − D
DESTINATION = FINZ
COMPTEUR = L − D + 1

Puis nous devons effectuer le transfert de haut en bas, soit de 1 à n (voir schéma de fonctionnement) ; l'instruction assembleur utilisée sera donc un LDDR.

La place étant alors libérée, la routine va pouvoir implanter le nouveau texte ; ce qui se fera par un transfert de bloc du nouveau texte à l'emplacement de l'ancien texte.

La routine va ensuite regarder si la fin de zone est atteinte. Si ce n'est pas fini, elle va procéder à une nouvelle recherche, et recommencer le processus de traitement exposé ci-dessus.

Routine effectuant le changement d'un texte par un autre, dans une liste



```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                  10 ;=* ROUTINE CHANGE POUR TOUS CPC *=
6FA0              20         ent  #6fa0
6FA0              30         org  #6fa0
                  40 ;--* TRAITEMENT RSX *--
BCD1              50 rsx:    equ  #bcd1
6FA0  01AA6F      60         ld   bc,ext
6FA3  21E46F      70         ld   hl,space
6FA6  CDD1BC      80         call rsx
6FA9  C9          90         ret
6FAA  DD6F       100 ext:    defw nom
                 110 ;--* RECUPERATION PARAMETRES *--
                 120 ;--* DEBUT DE ZONE *--
6FAC  DD6607     130         ld   h,(ix+7)
6FAF  DD6E06     140         ld   l,(ix+6)
                 150 ;--* FIN DE ZONE *--
6FB2  DD4605     160         ld   b,(ix+5)
6FB5  DD4E04     170         ld   c,(ix+4)
6FB8  ED433771   180         ld   (finz),bc
                 190 ;--* ADR TEXTE ANCIEN *--
6FBC  DD5603     200         ld   d,(ix+3)
6FBF  DD5E02     210         ld   e,(ix+2)
6FC2  ED531471   220         ld   (adrtxt),de
                 230 ;*-- ADR TEXTE NOUVEAU *--
6FC6  DD5601     240         ld   d,(ix+1)
6FC9  DD5E00     250         ld   e,(ix+0)
6FCC  ED533171   260         ld   (adrchg),de
6FD0  DD2A1471   270         ld   ix,(adrtxt)
6FD4  FD2A3171   280         ld   iy,(adrchg)
6FD8  FE04       290         cp   4
6FDA  C0         300         ret  nz
6FDB  1823       310         jr   rout
                 320 ;
6FDD  4348414E   330 nom:    defm "CHANG"
6FE2  C5         340         defb "E"+#80
6FE3  00         350         defb 0
6FE4             360 space:  defs 4
                 370 ;
                 380 ;--* DEBUT ROUTINE 'CHANGE' *--
7000             390         org  #7000
B900             400 romhau: equ  #b900
                 410 ;--* ROUTINE AFFICH. POUR CPC 6128 *--
C37D             420 affich: equ  #c37d
                 430 ;
7000  CD00B9     440 rout:   call romhau
                 450 ;--* COMPTEUR CAR. A 0 *--
```

```
7003  110000     460          ld   de,0
                 470 ;--* CAR. TEXTE ANC. DANS A *--
7006  DD7E00     480 lctl:    ld   a,(ix+0)
                 490 ;--* EST-CE FIN DE TEXTE ? *--
7009  FE00       500          cp   0
                 510 ;--* SI OUI FIN CALCUL LONG. *--
700B  2805       520          jr   z,ftxtl
                 530 ;--* +1 DANS COMPTEUR CAR. *--
700D  13         540          inc  de
                 550 ;--* +1 DANS POINTEUR TEXTE *--
700E  DD23       560          inc  ix
                 570 ;--* AUTRE CARACTERE *--
7010  18F4       580          jr   lctl
                 590 ;--* LONG TEXTE ANC DANS LTA *--
7012  ED533571   600 ftxtl:   ld   (lta),de
                 610 ;--* TEST SI LTA NUL *--
7016  7B         620          ld   a,e
7017  D600       630          sub  0
                 640 ;--* SI NUL MESSAGE ERREUR *--
7019  2841       650          jr   z,lgaz
                 660 ;--* REINIT IX AVEC ADR TEXTE *--
701B  DD2A1471   670          ld   ix,(adrtxt)
                 680 ;--* CALCUL LONG. TEXTE NOUV. *--
                 690 ;--* TRAIT. SIMIL. CI-DESSUS  *--
701F  110000     700          ld   de,0
7022  FD7E00     710 lct2:    ld   a,(iy+0)
7025  FE00       720          cp   0
7027  2805       730          jr   z,ftxt2
7029  13         740          inc  de
702A  FD23       750          inc  iy
702C  18F4       760          jr   lct2
702E  ED533371   770 ftxt2:   ld   (ltn),de
                 780 ;--* TEST SI LTN NUL *--
7032  7B         790          ld   a,e
7033  D600       800          sub  0
                 810 ;--* SI NUL MESSAGE ERREUR *--
7035  2825       820          jr   z,lgaz
7037  FD2A3171   830          ld   iy,(adrchg)
                 840 ;--* ADR TXT DANS IX *--
703B  DD2A1471   850 autre:   ld   ix,(adrtxt)
                 860 ;--* CAR TEXTE DANS A *--
703F  DD7E00     870          ld   a,(ix+0)
                 880 ;--* TEST A AVEC CAR ZONE *--
7042  BE         890 lcz:     cp   (hl)
7043  2048       900          jr   nz,difl
                 910 ;--* SAVE ADR 1ER CAR TROUVE *--
7045  221271     920          ld   (entier),hl
                 930 ;--* CAR TXT SUIV DANS A *--
7048  DD23       940 lch:     inc  ix
```

```
704A  DD7E00     950          ld   a,(ix+0)
                 960 ;--* TEST FIN TEXTE *--
704D  FE00       970          cp   0
704F  281A       980          jr   z,trouve
7051  CD9370     990          call finzon
7054  2805      1000          jr   z,ntrouv
7056  BE        1010          cp   (hl)
7057  28EF      1020          jr   z,lch
7059  182C      1030          jr   rech
705B  C9        1040 ntrouv: ret
                1050 ;--* AFF. MESS. LONG. TEXTE NUL *--
705C  214671    1060 lgaz:   ld   hl,mlgaz
705F  CD7DC3    1070         call affich
                1080 ;--* FIN DE TRAITEMENT *--
7062  C9        1090         ret
                1100 ;--* RECUP. HL  *--
7063  E1       1110 mfchg:  pop  hl
                1120 ;--* AFF. MESS. FIN DE ZONE *--
7064  211671    1130         ld   hl,txter
7067  CD7DC3    1140         call affich
706A  C9        1150         ret
                1160 ;--* SAUVE HL *--
706B  E5        1170 trouve: push hl
                1180 ;--* FINZ DANS HL *--
706C  C5        1190         push bc
706D  E1        1200         pop  hl
                1210 ;--* INIT COMPTEUR A 30 *--
706E  1E1E      1220         ld   e,30
                1230 ;--* 0 DANS A *--
7070  3E00      1240         ld   a,0
                1250 ;--* COMP. A ET CAR. POINTE PAR HL *--
7072  BE        1260 testz:  cp   (hl)
                1270 ;--* SI DIFF. ALORS FIN DE ZONE *--
7073  20EE      1280         jr   nz,mfchg
                1290 ;--* -1 DANS POINTEUR ZONE *--
7075  2B        1300         dec  hl
                1310 ;--* -1 DANS COMPTEUR *--
7076  1D        1320         dec  e
                1330 ;--* REITERER SI COMPT. DIFF. DE 0 *--
7077  20F9      1340         jr   nz,testz
                1350 ;--* RECUP HL *--
7079  E1        1360         pop  hl
                1370 ;--* SAUVE REGISTRES *--
707A  E5        1380         push hl
707B  C5        1390         push bc
707C  D5        1400         push de
                1410 ;--* CHANGER TEXTE ANC AVEC NOUV *--
707D  DDE5      1420         push ix
```

```
707F  CD9E70    1430          call change
                1440 ;--* RECUP REGISTRES *--
7082  DDE1      1450          pop  ix
7084  D1        1460          pop  de
7085  C1        1470          pop  bc
7086  E1        1480          pop  hl
                1490 ;--* RECH. AUTRE TEXTE *--
7087  CD9370    1500 rech:    call finzon
708A  20AF      1510          jr   nz,autre
708C  C9        1520          ret
708D  CD9370    1530 difl:    call finzon
7090  20B0      1540          jr   nz,lcz
7092  C9        1550          ret
                1560 ;--* +1 DANS PTR ZONE ET TEST SI FIN *--
                1570 ;--* PTR ZONE DANS DE *--
7093  23        1580 finzon: inc  hl
7094  E5        1590          push hl
7095  D1        1600          pop  de
                1610 ;--* SAVE PTR ZONE *--
7096  E5        1620          push hl
                1630 ;--* FIN ZONE DANS HL *--
7097  C5        1640          push bc
7098  E1        1650          pop  hl
                1660 ;--* CALCUL SI FIN DE ZONE *--
7099  A7        1670          and  a
709A  ED52      1680          sbc  hl,de
                1690 ;--* RECUP PTR ZONE *--
709C  E1        1700          pop  hl
709D  C9        1710          ret
                1720 ;--* AFF. MESS. CHANGE *--
709E  213D71    1730 change: ld   hl,messl
70A1  CD7DC3    1740          call affich
                1750 ;--* LTN DANS HL *--
70A4  2A3371    1760          ld   hl,(ltn)
                1770 ;--* LTA DANS DE *--
70A7  ED5B3571  1780          ld   de,(lta)
                1790 ;--* D = LTN - LTA *--
70AB  A7        1800          and  a
70AC  ED52      1810          sbc  hl,de
                1820 ;--* SAUVE D *--
70AE  E5        1830          push hl
                1840 ;--* FINZ DANS HL *--
70AF  C5        1850          push bc
70B0  E1        1860          pop  hl
                1870 ;--* POINTEUR TEXTE TROUVE DANS DE *--
70B1  ED5B1271  1880          ld   de,(entier)
                1890 ;--* CALCUL DE L DANS HL *--
70B5  A7        1900          and  a
70B6  ED52      1910          sbc  hl,de
```

```
70B8  ED5B3571  1920          ld   de,(lta)
                1930 ;--* L = FINZ - ENTIER - LTA *--
70BC  ED52      1940          sbc  hl,de
                1950 ;--* SAUVE L *--
70BE  E5        1960          push hl
                1970 ;--* POINTEUR TEXTE TROUVE DANS HL *--
70BF  2A1271    1980          ld   hl,(entier)
                1990 ;--* LTA DANS DE *--
70C2  ED5B3571  2000          ld   de,(lta)
                2010 ;--* CALCUL SOURCE POUR TRANSFERT *--
70C6  A7        2020          and  a
70C7  ED5A      2030          adc  hl,de
                2040 ;--* SOURCE = ENTIER + LTA *--
70C9  223971    2050          ld   (source),hl
                2060 ;--* POINTEUR TEXTE TROUVE DAND HL *--
70CC  2A1271    2070          ld   hl,(entier)
                2080 ;--* LTN DANS DE *--
70CF  ED5B3371  2090          ld   de,(ltn)
                2100 ;--* DEST = ENTIER + LTN *--
70D3  A7        2110          and  a
70D4  ED5A      2120          adc  hl,de
70D6  223B71    2130          ld   (dest),hl
                2140 ;--* RECUP L DANS HL *--
70D9  E1        2150          pop  hl
                2160 ;--* RECUP D DANS DE *--
70DA  D1        2170          pop  de
                2180 ;--* TEST SIGNE DE D *--
70DB  CB7A      2190          bit  7,d
                2200 ;--* ALLER A DNEG SI NEGATIF *--
70DD  2016      2210          jr   nz,dneg
                2220 ;--* L1 = L - D (DANS HL) *--
70DF  A7        2230          AND  A
70E0  ED52      2240          SBC  HL,DE
                2250 ;--* L1 DANS DE *--
70E2  E5        2260          push hl
70E3  D1        2270          pop  de
                2280 ;--* SAUVE L1 *--
70E4  E5        2290          push hl
                2300 ;--* RECUP SOURCE DANS HL *--
70E5  2A3971    2310          ld   hl,(source)
                2320 ;--* SOURCE = ENTIER + LTA + L1 *--
70E8  A7        2330          and  a
70E9  ED5A      2340          adc  hl,de
                2350 ;--* DEST = FINZ *--
70EB  ED5B3771  2360          ld   de,(finz)
                2370 ;--* RECUP L1 DANS BC *--
70EF  C1        2380          pop  bc
                2390 ;--* +1 DANS L1 *--
```

```
70F0  03        2400          inc  bc
                2410 ;--* TRANSFERT BLOC POUR D POSITIF *--
70F1  EDB8      2420          lddr
70F3  1805      2430          jr   chgtxt
                2440 ;--* APPEL PARM POUR TRANSFERT *--
70F5  CD0871    2450 dneg:    call csd
                2460 ;--* TRANSFERT BLOC POUR D NEGATIF *--
70F8  EDB0      2470          ldir
                2480 ;--* METTRE NOUVEAU TEXTE *--
                2490 ;--* LTN DANS BC (COMPTEUR) *--
70FA  ED4B3371  2500 chgtxt: ld   bc,(ltn)
                2510 ;--* POINTEUR NOUV TEXTE DANS HL *--
70FE  2A3171    2520          ld   hl,(adrchg)
                2530 ;--* POINTEUR TEXTE ANC DANS DE *--
7101  ED5B1271  2540          ld   de,(entier)
                2550 ;--* TRANSFERT NOUVEAU TEXTE *--
7105  EDB0      2560          ldir
7107  C9        2570          ret
                2580 ;--* RECUP LONGUEUR DANS BC *--
7108  E5        2590 csd:     push hl
7109  C1        2600          pop  bc
                2610 ;--* SOURCE DANS HL *--
710A  2A3971    2620          ld   hl,(source)
                2630 ;--* DEST DANS DE *--
710D  ED5B3B71  2640          ld   de,(dest)
7111  C9        2650          ret
                2660 ;
                2670 ;--* ZONES UTILISEES PAR LA ROUTINE *--
                2680 ;
                2690 ;--* POINTEUR TEXTE TROUVE *--
7112            2700 entier: defs 2
                2710 ;--* POINTEUR TEXTE ANCIEN *--
7114            2720 adrtxt: defs 2
                2730 ;--* MESSAGE CHANGE IMPOSSIBLE *--
7116  66696E20  2740 txter:  defm "fin de zone texte atteint"
712F  0A00      2750 saut:   defb 10,0
                2760 ;--* POINTEUR TEXTE NOUVEAU *--
7131            2770 adrchg: defs 2
                2780 ;--* LONG. TEXTE NOUVEAU *--
7133            2790 ltn:    defs 2
                2800 ;--* LONG. TEXTE ANCIEN *--
7135            2810 lta:    defs 2
                2820 ;--* POINTEUR FIN DE ZONE *--
7137            2830 finz:   defs 2
                2840 ;--* SOURCE POUR TRANSFERT BLOC *--
7139            2850 source: defs 2
                2860 ;--* DEST POUR TRANSFERT BLOC *--
713B            2870 dest:   defs 2
                2880 ;--* MESSAGE CHANGE EFFECTUE *--
```

```
713D  6368616E  2890 messl:  defm "change "
7144  0A00      2900 sautl:  defb 10,0
                2910 ;--* MESSAGE ERREUR ENTREE / ARRET *--
7146  6C6F6E67  2920 mlgaz:  defm "longueur texte(s) nul - fin trait."
7168  0A00      2930 saut2:  defb 10,0

Pass 2 errors: 00

adrchg 7131  adrtxt 7114  affich C37D
autre  703B  change 709E  chgtxt 70FA
csd    7108  dest   713B  difl   708D
dneg   70F5  entier 7112  ext    6FAA
finz   7137  finzon 7093  ftxtl  7012
ftxt2  702E  lch    7048  lctl   7006
lct2   7022  lcz    7042  lgaz   705C
lta    7135  ltn    7133  messl  713D
mfchg  7063  mlgaz  7146  nom    6FDD
ntrouv 705B  rech   7087  romhau B900
rout   7000  rsx    BCD1  saut   712F
sautl  7144  saut2  7168  source 7139
space  6FE4  testz  7072  trouve 706B
txter  7116

Table used:   478  from   900
Executes: 28576
```

```
10 REM === DEMONSTRATION ROUTINE 'CHANGE' ===
20 MODE 2
30 PRINT"init zone a zero"
40 FOR i=&9000 TO &9A00:POKE i,0:NEXT
50 PRINT"chargement texte":PRINT
60 j=&9000
70 READ a$
80 IF a$="fin" THEN 160
90 l=LEN(a$)
100 FOR i=1 TO l
110 c$=MID$(a$,i,1):c=ASC(c$)
120 PRINT c$;:POKE j+i-1,c
130 NEXT
140 PRINT:j=j+l
150 GOTO 70
160 LOCATE 25,25:PRINT"TAPER UNE TOUCHE
POUR CONTINUER"
170 CALL &BB06:CLS
180 INPUT "texte a chercher:";a$
190 INPUT "texte a changer:";b$
200 l=LEN(a$):at=&8000
210 FOR i=1 TO l
220 c$=MID$(a$,i,1):c=ASC(c$)
230 POKE at+i-1,c
240 NEXT
250 POKE at+i-1,0
260 l=LEN(b$):at=&8100
270 FOR i=1 TO l
280 c$=MID$(b$,i,1):c=ASC(c$)
290 POKE at+i-1,c
300 NEXT
310 POKE at+i-1,0
320 |CHANGE,&9000,&9A00,&8000,&8100
330 j=&9000:a$="":PRINT
340 v=PEEK(j)
350 IF v=0 THEN 400
360 c$=CHR$(v):a$=a$+c$:cpt=cpt+1:j=j+1
370 IF cpt < 80 THEN 340
380 PRINT a$:cpt=0:a$=""
390 GOTO 340
400 IF cpt < 80 THEN PRINT a$
410 PRINT
420 INPUT "autre change (o/n):";r$:r$=LO
WER$(r$)
430 IF r$="o" THEN cpt=0:GOTO 160
440 END
450 DATA "une programmation en langage m
achine apporte dans la majorite des cas
"
```

```
460 DATA "une puissance d'execution que
nul autre langage ne permet. "
470 DATA "tous les autres langages neces
sitent une ou plusieurs traductions "
480 DATA "avant d'etre directement execu
table par le microprocesseur. "
490 DATA "le seul langage disponible imm
ediatement lors du branchement de votre
"
500 DATA "amstrad cpc est un langage bas
ic resident. "
510 DATA "fin"
```

ROUTINES DE TRI DE TABLEAUX

Qui n'a pas été confronté au problème crucial du tri en mémoire ! Ce problème est souvent difficile à résoudre au niveau du BASIC. En effet, ou le tri réalisé prend peu de place mémoire mais s'avère lent, ou au contraire il peut être rapide mais très gourmand en place mémoire, car nécessitant plusieurs tableaux de travail en mémoire.

Nous sommes donc irrémédiablement contraints de nous tourner vers le langage machine, qui nous offre la rapidité de calcul que ne possède pas le BASIC.
C'est ce que nous avons fait dans cette troisième section de ce chapitre sur le traitement des listes.
Nous avons pensé que le tri de tableaux définis en BASIC, par l'instruction DIM, s'avérait limitatif.
Aussi avons-nous réalisé des routines permettant de réaliser le tri de tableaux créés directement par l'utilisateur, c'est-à-dire des tableaux qui ne sont pas définis par une instruction DIM.
Toutefois les tableaux ainsi créés auront une structure identique aux tableaux définis en DIM, de façon que les routines présentées tournent différemment sur les deux types de tableaux.

Mais avant de nous lancer dans le traitement des routines, il nous faut avant tout savoir comment sont organisés les tableaux en mémoire de l'Amstrad CPC.

L'Amstrad nous offre trois sortes de tableaux :

— les tableaux de chaînes (exemple : T$(10))
— les tableaux de réels (exemples : T(10))
— les tableaux d'entiers (exemple : t%(10))

Nous allons examiner la structure d'un tableau en général, mais les routines présentées ne traiteront que les tableaux à une dimension ; en effet, le tri d'un tableau à plusieurs dimensions n'a souvent pas de sens.
Si le cas se présentait, il serait très facile de le ramener à un ensemble de plusieurs tableaux à une dimension.

Mais avant d'étudier la structure d'un tableau en mémoire, il faut connaître son emplacement en mémoire.
L'Amstrad CPC est généreux sur ce point, puisqu'il nous offre des pointeurs de début et de fin de tableaux.

Pointeurs de tableaux :

POINTEURS	CPC 464	CPC 664 & 6128
début de tableaux	#AE87 — #AE88	#AE6A — #AE6B
fin de tableaux	#AE89 — #AE8A	#AE6C — #AE6D

Structure d'un tableau :

ELEMENTS DU TABLEAU	NB OCTETS
valeur variable	1
valeur à zéro	1
nom du tableau (en ASCII)	long. nom
type de variable du tableau	1
taille du tableau	2
nombre d'indices du tableau	1
nombre de postes des N indices	N fois 2
si entier →	X fois 2
X éléments du tableau si réel →	X fois 5
si chaîne →	X fois 3

Précisions concernant les éléments du tableau :

Nom du tableau :

Le code ASCII du dernier caractère du nom est ajouté à la valeur #80 pour que le système sache reconnaître la fin du nom.

Type de variable :

Variable entière (en %) : 1
Variable chaîne (en $) : 2
Variable réelle : 4

Taille du tableau :

C'est la longueur entre le nombre d'indice et le dernier élément du tableau, les deux étant compris dans la longueur déterminée. C'est aussi le résultat du calcul suivant :

taille = (nb postes indice 1 x... x nb postes indice N) + (nb indices × 2) + 1

Eléments du tableau :

Nous avons vu que les éléments pouvaient avoir une longueur différente selon que le type de variable indique de l'entier, du réel, ou un type chaîne.

Dans le cas d'entier ou de réel, il s'agit de la valeur elle-même qui est présente dans le tableau, donc sur 2 ou 5 octets.

Par contre, dans le cas de variables chaîne, nous avons 3 octets, dont la signification est la suivante :

PREMIER OCTET : nombre de caractères de la chaîne
2e et 3e OCTETS : adresse d'implantation de la chaîne en mémoire RAM de l'Amstrad CPC.

L'implantation d'une chaîne peut se situer n'importe où dans le programme BASIC :
— en DATA (exemple : DATA wxyz)
— dans une variable chaîne (exemple : A$ = « wxyz »)
— en remplissant un tableau (exemple : T$(4) = « wxyz »)

De toute manière, vous n'avez pas à vous soucier de ce problème puisque c'est le système qui le gère.

Nous avons vu dans les pointeurs qu'il existait un pointeur de début de tableau ; en fait, ce pointeur est peu intéressant dans notre cas, car il indique l'adresse de début du PREMIER tableau. Or, si nous devons trier un tableau qui se situe en Nième position dans une instruction DIM, il nous faudra effectuer de savants calculs pour déterminer l'adresse de début du tableau voulu.

Cependant nous disposons, en BASIC, d'une possibilité très intéressante, l'arobase "@".
En effet, si vous faites par exemple PRINT @X, vous obtiendrez l'adresse d'implantation de la variable X.
En ce qui concerne les tableaux, nous pouvons appliquer le même principe, en faisant par exemple PRINT @T%(0), qui nous donnera l'adresse du premier élément du tableau T%.
Aussi nous utiliserons ce principe pour passer l'adresse de début du premier élément à la routine, par l'intermédiaire d'une RSX, ou d'une instruction Basic CALL (voir DEMO).

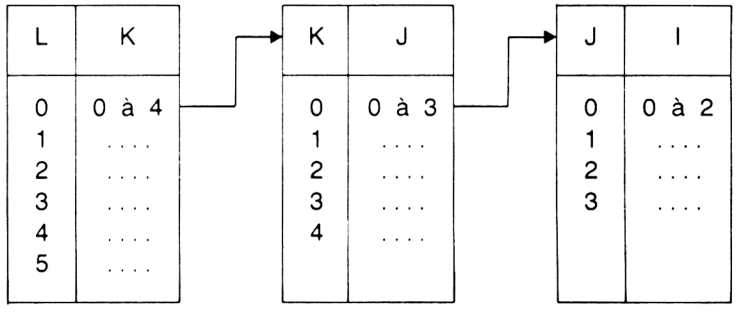
Exemple de structure d'un tableau :

Soit le tableau entier à 4 dimensions : ABCD%(2,3,4,5) indices → I J K L

00 00	premier octet du tableau deuxième .....
41 42 43 43 + 80	premier caractère ASCII du tableau (en hexa) deuxième ..... troisième ..... dernier caractère ASCII (valeur hexa + #80)
02	type de variable : entier
D9 02	octet faible de la taille du tableau octet fort.....
04	nombre d'indices du tableau
06 00	octet faible du nb de postes de l'indice L octet fort.....
05 00	octet faible du nb de postes de l'indice K octet fort.....
04 00	octet faible du nb de postes de l'indice J octet fort.....
03 00	octet faible du nb de postes de l'indice I octet fort.....
CONTENU DU TABLEAU : valeurs entières	

Structure des valeurs d'un tableau :

Reprenons notre exemple T%(2,3,4,5)
indices → I J K L



Ce schéma demande quelques explications :

Pour une valeur de L = 0 (par exemple), nous avons K qui prend les valeurs 0 à 4, et dans cet ordre ; nous dirons que L est hiérarchiquement supérieur à K.

Puis pour toute valeur de L, il en sera de même ; ce qui implique que nous aurons une même structure pour chaque valeur de L.

De même, pour chaque valeur de K, nous avons des groupes de J, qui vont de la valeur 0 à la valeur 3 ; ici également, K englobe hiérarchiquement l'ensemble des J. Puis de la même façon, chaque J se trouve à son tour supérieur hiérarchiquement à I.

Nous allons en définitive trouver, dans le tableau, un certain nombre de groupes de I, allant de 0 à 2.
Pour être plus précis, nous en trouverons 3 pour chaque J, soit 12 pour chaque K, soit 60 pour chaque L.

Deux remarques importantes sont donc à noter au passage :

(1) tout indice part de 0 (ex. : pour un indice égal à 2 nous avons 3 postes qui vont de 0 à 2).

(2) l'indice le plus haut dans la hiérarchie est le dernier,
l'indice le plus bas dans la hiérarchie est le premier.

La progression des postes, en mémoire, se fait donc toujours du poste 0 au poste N.

ROUTINE DE TRI DE TABLEAU DE CHAÎNES

Méthode de tri utilisée

La méthode de tri qui sera utilisée pour les trois routines de tri sera le « TRI BULLE », en optant pour un classement croissant des éléments du tableau à trier.
Cette méthode de tri est la plus simple à mettre en œuvre, et ne nécessite aucune place mémoire, en dehors du tableau à trier.

Le principe en est très simple, puisqu'il s'agit de comparer les éléments 2 par 2, en partant du début du tableau, et de mettre le plus grand des 2 dans l'élément supérieur, par inversion de ces 2 éléments ; arrivé à la fin du tableau, le dernier élément doit donc être le plus grand du tableau.
Au deuxième passage, nous effectuons un tri du premier à l'avant dernier élément, en utilisant le même processus de tri.
Puis nous continuons ainsi jusqu'à ce que le tri des éléments se limite à 1 seul élément ; le tableau est alors trié.

Explications sur la routine de tri

Nous avons vu, dans la structure des tableaux de chaînes, que chaque élément était en fait constitué d'une longueur et d'une adresse de chaîne.
Le tri de ces chaînes s'effectuera donc, en réalité, sur les éléments eux-mêmes.
Pour ce faire, nous allons comparer la première chaîne pointée par l'adresse du premier élément à la deuxième chaîne pointée par l'adresse du deuxième élément ; il sera tenu compte des longueurs fournies par ces deux éléments.

Si la deuxième chaîne est plus grande que la première, il nous faudra alors effectuer une inversion des longueurs et adresses de ces deux éléments.

Le processus sera le même jusqu'à la fin, et le tableau ainsi trié offrira en fait un classement sur les longueurs et adresses. La comparaison de deux chaînes se fera caractère par caractère. Si une égalité se produit, nous irons alors comparer les deux caractères suivants, dans le cas ou les deux chaînes ne sont pas terminées ; sinon nous irons regarder si la première chaîne est terminée, auquel cas nous n'effectuerons pas d'inversion.

Si par contre c'est la deuxième qui est terminée, nous irons effectuer l'inversion des 2 éléments en cours.

Si les deux caractères comparés sont différents, il nous faudra alors savoir lequel des deux est le plus grand.
S'il s'agit du caractère de la première chaîne, nous devrons alors effectuer l'inversion des 2 éléments en cours.
Par contre, s'il s'agit du caractère de la deuxième chaîne, nous n'effectuerons alors aucune inversion.

Démonstration de la routine de tri

Deux courts programmes Basic vous montreront les possibilités offertes par cette routine de tri.

Le premier programme effectue le tri d'un tableau défini en DIM.

Le deuxième programme, quant à lui, effectue le tri d'un tableau qui a été créé de toutes pièces.
Les chaînes à trier sont mises à un emplacement mémoire choisi par le créateur du tableau.

Pour constituer l'ensemble des éléments du tableau, il nous faut repérer chaque adresse de début de chaîne, ainsi que sa longueur. Vous remarquerez également, dans le programme BASIC, qu'il nous faut connaître l'adresse du premier élément du tableau.

Par ailleurs, il ne faut pas oublier de mettre les 4 données qui précèdent le premier élément, soit le type de variable, la taille du tableau (qui sera mise à 0, car inutilisée dans la routine), le nombre d'indices (soit 1), et le nombre de postes de l'indice.

Une possibilité intéressante offerte par la routine est le choix du nombre des premiers éléments à trier.
Il vous sera ainsi possible de ne trier qu'une partie de votre tableau, mais à partir du premier élément.

L'instruction RSX correspondante est la suivante :

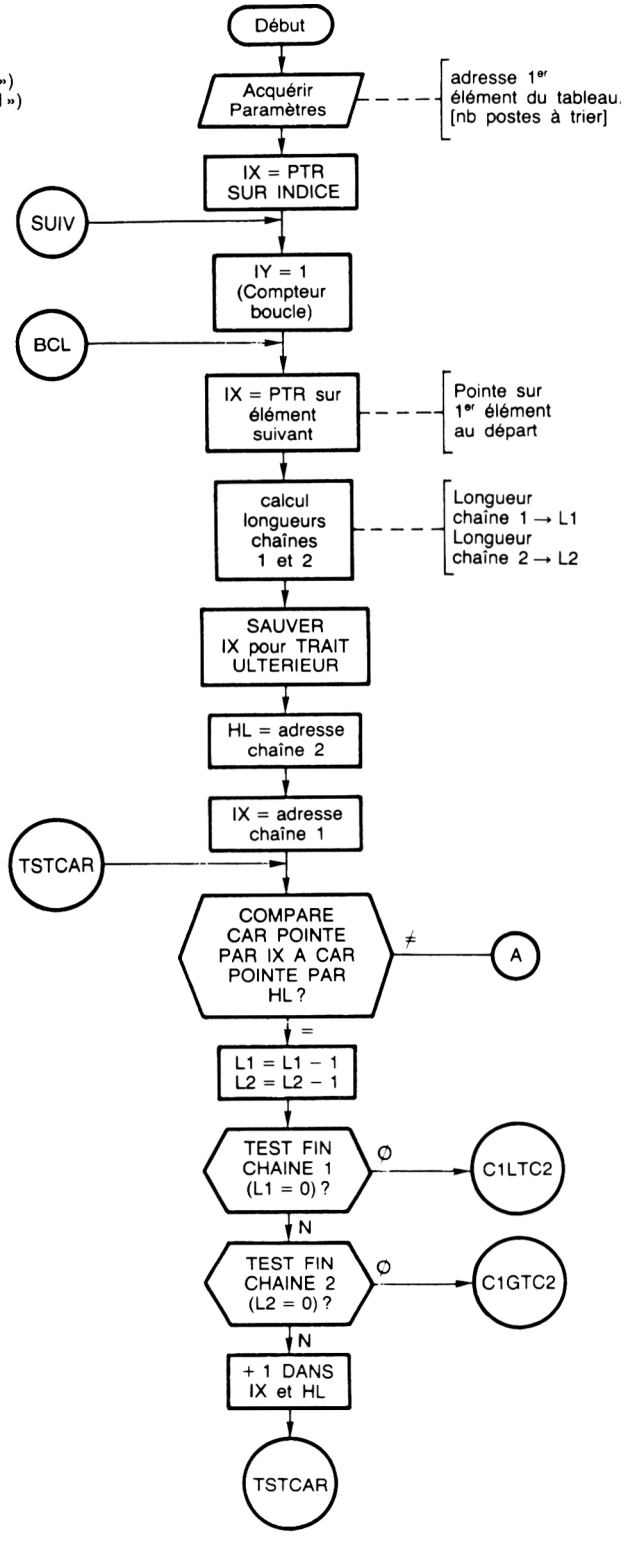
|TRIC, adresse du 1er élément, [nombre d'éléments à trier]

Si vous n'indiquez pas le nombre d'élément à trier, ce sera alors tout le tableau qui sera trié.

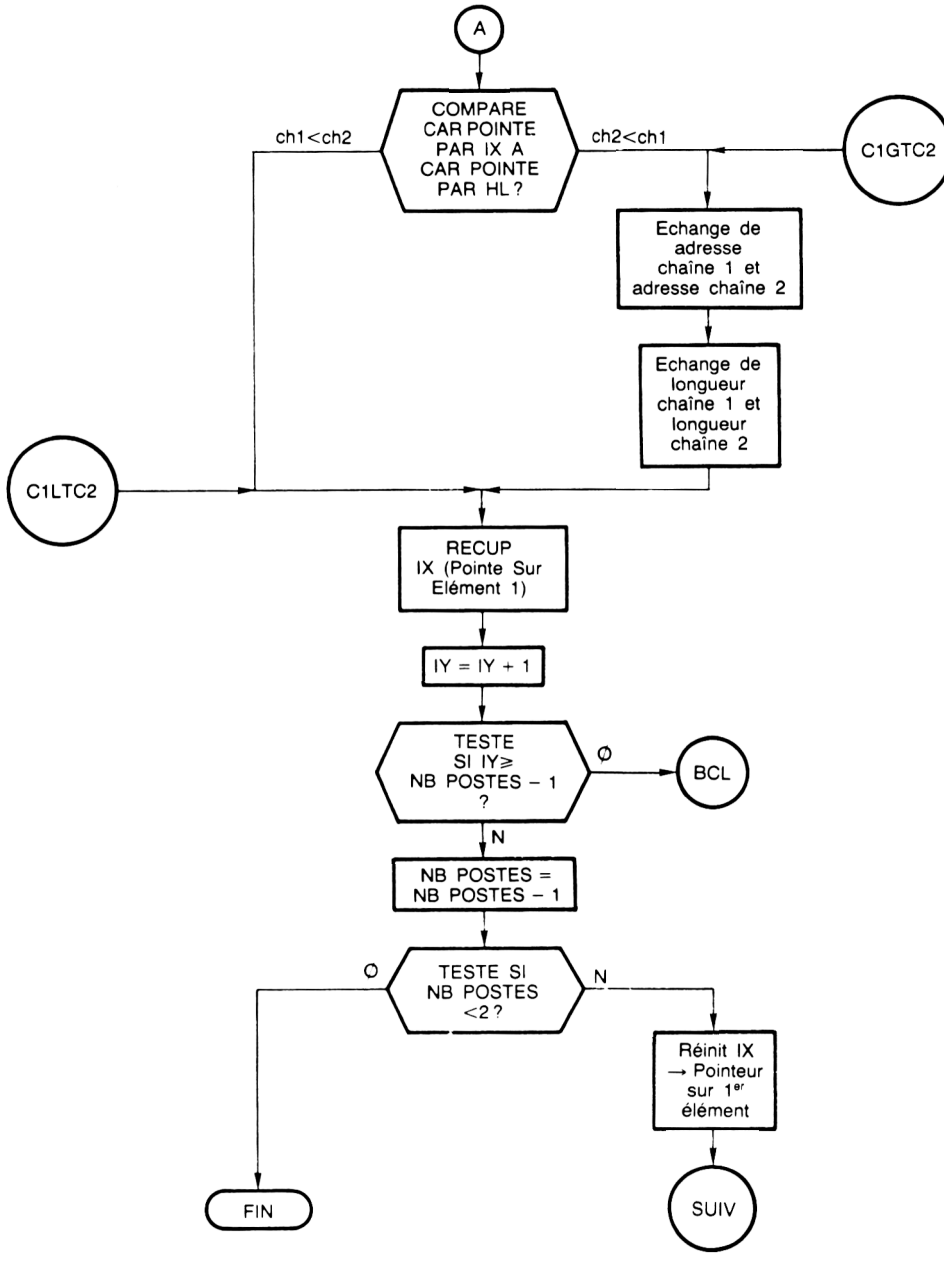
Routine de tri de tableau de chaînes

Légende :

0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → lettre (pour « NON »)



Routine de tri de tableau de chaînes (suite)



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                 10 ;=* ROUTINE TRI DE TABLEAU DE CHAINE *=
6FA0             20         ent  #6fa0
6FA0             30         org  #6fa0
                 40 ;--* TRAITEMENT RSX *--
BCD1             50 rsx:    equ  #bcd1
6FA0  01AA6F     60         ld   bc,ext
6FA3  21D36F     70         ld   hl,space
6FA6  CDD1BC     80         call rsx
6FA9  C9         90         ret
6FAA  CE6F      100 ext:    defw nom
                110 ;--* RECUPERATION PARAMETRES *--
                120 ;--* HL=deb tabl / BC=nb elem a trier *--
6FAC  FE01      130         cp   1
6FAE  2805      140         jr   z,un
6FB0  FE02      150         cp   2
6FB2  280C      160         jr   z,deux
6FB4  C9        170         ret
                180 ;--* TRAIT 1 PARAMETRE *--
6FB5  010000    190 un:     ld   bc,0
6FB8  DD6601    200         ld   h,(ix+1)
6FBB  DD6E00    210         ld   l,(ix+0)
6FBE  1840      220         jr   rout
                230 ;--* TRAIT 2 PARAMETRES *--
6FC0  DD6603    240 deux:   ld   h,(ix+3)
6FC3  DD6E02    250         ld   l,(ix+2)
6FC6  DD4601    260         ld   b,(ix+1)
6FC9  DD4E00    270         ld   c,(ix+0)
6FCC  1832      280         jr   rout
                290 ;
6FCE  545249    300 nom:    defm "TRI"
6FD1  C3        310         defb "C"+#80
6FD2  00        320         defb 0
6FD3            330 space:  defs 4
                340 ;
                350 ;--* DEBUT ROUTINE 'TRIC' *--
7000            360         org  #7000
B900            370 romhau: equ  #b900
C37D            380 affich: equ  #c37d
7000  CD00B9    390 rout:   call romhau
                400 ;--* PTR HL SUR NB POSTES *--
7003  2B        410         dec  hl
7004  2B        420         dec  hl
                430 ;--* SAUVE PTR HL DANS IX *--
7005  E5        440         push hl
```

```
7006  DDE1       450          pop  ix
                 460 ;--* TEST SI NB POSTES NUL *--
7008  0B         470          dec  bc
7009  CB78       480          bit  7,b
700B  2808       490          jr   z,post
                 500 ;--* SI NUL TRIER TOUS *--
700D  DD4601     510          ld   b,(ix+1)
7010  DD4E00     520          ld   c,(ix+0)
7013  181A       530          jr   postok
                 540 ;--* RECTIF NB POSTES *--
7015  03         550 post:    inc  bc
7016  CB78       560          bit  7,b
                 570 ;--* SI NEGATIF ARRET *--
7018  2815       580          jr   z,postok
701A  215771     590          ld   hl,mess3
701D  CD7DC3     600          call affich
7020  C9         610          ret
7021  210871     620 merr1:   ld   hl,mess1
7024  CD7DC3     630          call affich
7027  C9         640          ret
7028  212F71     650 merr2:   ld   hl,mess2
702B  CD7DC3     660          call affich
702E  C9         670          ret
                 680 ;--* PTR HL SUR NB INDICES *--
702F  2B         690 postok: dec  hl
                 700 ;--* TEST SI PLUS D'UN INDICE *--
7030  3E01       710          ld   a,1
7032  BE         720          cp   (hl)
                 730 ;--* SI ERREUR ARRET *--
7033  20EC       740          jr   nz,merr1
                 750 ;--* PTR HL SUR TYPE VAR *--
7035  2B         760          dec  hl
7036  2B         770          dec  hl
7037  2B         780          dec  hl
                 790 ;--* TEST SI TYPE VAR = CHAINE *--
7038  3E02       800          ld   a,2
703A  BE         810          cp   (hl)
                 820 ;--* SI NON CHAINE ARRET *--
703B  20EB       830          jr   nz,merr2
                 840 ;--* PTR IX SUR NB INDICES *--
703D  DD2B       850          dec  ix
                 860 ;--* SAUVE PTR IX ET NB POSTES *--
703F  DD220671   870          ld   (ptrorg),ix
7043  ED43FE70   880          ld   (nbpost),bc
                 890 ;--* CPT K = 1 *--
7047  FD210100   900 suiv:    ld   iy,1
                 910 ;--* PTR IX SUR PROCH. ELEM. *--
704B  DD23       920 bcl:     inc  ix
704D  DD23       930          inc  ix
```

```
704F  DD23       940          inc  ix
                 950 ;--* SAUVE LONG ELEMENT 1 *--
7051  DD7E03     960          ld   a,(ix+3)
7054  320471     970          ld   (l2),a
                 980 ;--* SAUVE LONG ELEMENT 2 *--
7057  DD7E00     990          ld   a,(ix+0)
705A  320271    1000          ld   (l1),a
                1010 ;--* SAUVE PTR IX *--
705D  DDE5      1020          push ix
                1030 ;--* ADR CHAINE 2 DANS HL *--
705F  DD6E04    1040          ld   l,(ix+4)
7062  DD6605    1050          ld   h,(ix+5)
                1060 ;--* ADR CHAINE 1 DANS IX *--
7065  DD5E01    1070          ld   e,(ix+1)
7068  DD5602    1080          ld   d,(ix+2)
706B  D5        1090          push de
706C  DDE1      1100          pop  ix
                1110 ;--* CAR CHAINE 1 POINTE PAR IX *--
706E  DD7E00    1120 tstcar: ld   a,(ix+0)
                1130 ;--* COMPARE A ET CAR CHAINE 2 *--
7071  BE        1140          cp   (hl)
                1150 ;--* SI EGAL ALLER A MEMCAR *--
7072  2809      1160          jr   z,memcar
                1170 ;--* COMPARE CAR CH1 A CAR CH2 *--
7074  5E        1180          ld   e,(hl)
7075  A7        1190          and  a
7076  93        1200          sub  e
7077  CB7F      1210          bit  7,a
                1220 ;--* ALLER A C1LTC2 SI INFERIEUR *--
7079  205D      1230          jr   nz,c1ltc2
                1240 ;--* SINON ALLER A C1GTC2 *--
707B  182B      1250          jr   c1gtc2
707D  ED5B0271  1260 memcar: ld   de,(l1)
7081  1B        1270          dec  de
7082  ED530271  1280          ld   (l1),de
7086  CB7A      1290          bit  7,d
                1300 ;--* ALLER A C1LTC2 SI FIN CH1 *--
7088  204E      1310          jr   nz,c1ltc2
708A  ED5B0471  1320          ld   de,(l2)
708E  1B        1330          dec  de
708F  ED530471  1340          ld   (l2),de
7093  CB7A      1350          bit  7,d
                1360 ;--* ALLER A C1GTC2 SI FIN CH2 *--
7095  2011      1370          jr   nz,c1gtc2
                1380 ;--* INCR PTR CH1 ET CH2 *--
7097  DD23      1390          inc  ix
7099  23        1400          inc  hl
                1410 ;--* ALLER A TSTCAR SI MEME CAR *--
```

```
709A  18D2      1420          jr   tstcar
                1430 ;--* RECUP K ET VA TRAIT COUPLE SUIV *--
709C  FDE1      1440 pbcl:    pop  iy
709E  18AB      1450          jr   bcl
                1460 ;--* RECUP K ET PTR IX 1ER ELEM *--
70A0  FDE1      1470 psuiv:   pop  iy
70A2  DD2A0671  1480          ld   ix,(ptrorg)
                1490 ;--* CONT ISOLER ELEM SUPERIEUR *--
70A6  189F      1500          jr   suiv
                1510 ;--* TRAIT SI CAR CH1 SUP A CAR CH2 *--
70A8  DDE1      1520 clgtc2: pop   ix
70AA  DDE5      1530          push ix
                1540 ;--* ECHANGE ADR CH1 ET ADR CH2 *--
70AC  DD5602    1550          ld   d,(ix+2)
70AF  DD5E01    1560          ld   e,(ix+1)
70B2  ED530071  1570          ld   (temp),de
70B6  DD5605    1580          ld   d,(ix+5)
70B9  DD5E04    1590          ld   e,(ix+4)
70BC  DD7202    1600          ld   (ix+2),d
70BF  DD7301    1610          ld   (ix+1),e
70C2  ED5B0071  1620          ld   de,(temp)
70C6  DD7205    1630          ld   (ix+5),d
70C9  DD7304    1640          ld   (ix+4),e
                1650 ;--* ECHANGE LONG CH1 ET LONG CH2 *--
70CC  DD7E00    1660          ld   a,(ix+0)
70CF  DD5E03    1670          ld   e,(ix+3)
70D2  DD7300    1680          ld   (ix+0),e
70D5  DD7703    1690          ld   (ix+3),a
                1700 ;--* RECUP IX POUR TRAIT C1 LT C2 *--
70D8  DDE1      1710 c1ltc2: pop   ix
                1720 ;--* RECUP K DANS DE *--
70DA  FDE5      1730          push iy
70DC  D1        1740          pop  de
                1750 ;--* K = K + 1 *--
70DD  13        1760          inc  de
                1770 ;--* SAUVE K *--
70DE  D5        1780          push de
                1790 ;--* RECUP NB POSTES (J) *--
70DF  2AFE70    1800          ld   hl,(nbpost)
                1810 ;--* J = J - 1 *--
70E2  2B        1820          dec  hl
70E3  A7        1830          and  a
70E4  ED52      1840          sbc  hl,de
70E6  CB7C      1850          bit  7,h
                1860 ;--* ALLER A PBCL SI K SUP OU EGAL A J-1 *--
70E8  28B2      1870          jr   z,pbcl
                1880 ;--* SINON J=J-1 *--
70EA  2AFE70    1890          ld   hl,(nbpost)
```

```
70ED  2B        1900          dec  hl
70EE  22FE70    1910          ld   (nbpost),hl
70F1  110200    1920          ld   de,2
70F4  A7        1930          and  a
70F5  ED52      1940          sbc  hl,de
70F7  CB7C      1950          bit  7,h
                1960 ;--* SI J SUP OU EGAL A 2 ALLER A PSUIV *--
70F9  28A5      1970          jr   z,psuiv
                1980 ;--* REEQUILIBRE PILE *--
70FB  FDE1      1990          pop  iy
                2000 ;--* FIN SI J INFERIEUR A 2 *--
70FD  C9        2010          ret
                2020 ;
                2030 ;--* ZONES UTILISEES POUR DONNEES *--
70FE            2040 nbpost: defs 2                    ;NB POSTES TABL
7100            2050 temp:   defs 2                    ;ZONE D'ECHANGE
7102            2060 l1:     defs 2                    ;LG CHAINE 1
7104            2070 l2:     defs 2                    ;LG CHAINE 2
7106            2080 ptrorg: defs 2                    ;PTR 1ER ELEM
                2090 ;--* MESSAGE ERREUR AVANT ARRET *--
7108  6C652074  2100 mess1:  defm "le tableau a plus d'un indice / arret"
712D  0A00      2110 saut1:  defb 10,0
712F  6C652074  2120 mess2:  defm "le tableau n'est pas en 'nom$' / arret"
7155  0A00      2130 saut2:  defb 10,0
7157  6E622065  2140 mess3:  defm "nb elements a trier incorrect / arret"
717C  0A00      2150 saut3:  defb 10,0


Pass 2 errors: 00


affich C37D  bcl    704B  clgtc2 70A8
clltc2 70D8  deux   6FC0  ext    6FAA
l1     7102  l2     7104  memcar 707D
merr1  7021  merr2  7028  mess1  7108
mess2  712F  mess3  7157  nbpost 70FE
nom    6FCE  pbcl   709C  post   7015
postok 702F  psuiv  70A0  ptrorg 7106
romhau B900  rout   7000  rsx    BCD1
saut1  712D  saut2  7155  saut3  717C
space  6FD3  suiv   7047  temp   7100
tstcar 706E  un     6FB5
```

Démo de tri d'un tableau de chaîne T$ (20)

```
10 MODE 2
20 DIM t$(20)
30 CLS
40 FOR i=0 TO 20
50 READ a$
60 PRINT a$
70 t$(i)=a$
80 NEXT
90 LOCATE 25,25:PRINT"APPUYEZ SUR UNE TO
UCHE POUR CONTINUER":CALL &BB06
100 PRINT
110 a%=àt$(0)
120 |TRIC,a%
130 PRINT
140 FOR i=0 TO 20
150 PRINT t$(i)
160 NEXT
170 PRINT
180 END
190 DATA salut,pomme,disque,crayon,papie
r,arbre,travail,arret,double,arc,arme,ar
mee,salutation,travailler,abcde,abcd,abc
,a,zut,zoo,pole
```

Démo de tri d'une zone d'adresses de chaîne

```
10 FOR i=1 TO 24
20 READ v:POKE &14F+i,v
30 NEXT
40 FOR i=1 TO 6
50 READ a$:l=LEN(a$)
60 FOR j=1 TO l
65 c$=MID$(a$,j,1):c=ASC(c$)
70 POKE 299+ptr+j,c
80 NEXT
90 ptr=ptr+l
100 NEXT
105 PRINT"fin chargement"
110 a=342
120 |TRIC,a
130 END
140 DATA 2,0,0,1,6,0,4,&2c,1,5,&31,1,7,&
37,1,6,&3f,1,3,&46,1,6,&4a,1
150 DATA pole.,pomme.,abricot.,disque.,r
at.,orange
```

ROUTINE DE TRI DE TABLEAU D'ENTIERS

Les éléments du tableau à trier sont directement contenus dans le tableau (valeurs entières signées sur 2 octets).
Le processus de tri est le même que pour le tri de tableau de chaînes, mais la procédure de comparaison des éléments s'en trouve simplifiée.
En effet, si le premier entier est plus grand que le second, nous effectuons alors une inversion des deux éléments, sinon rien.

Démonstration de la routine de tri

Deux programmes BASIC sont également présentés pour cette routine de tri de tableau d'entiers.
Le premier programme trie un tableau défini en DIM, alors que le deuxième tri des valeurs qui ont été mises à un endroit de la mémoire RAM.
Ici également, il ne faut pas oublier de mettre les 4 données précédant le premier entier.

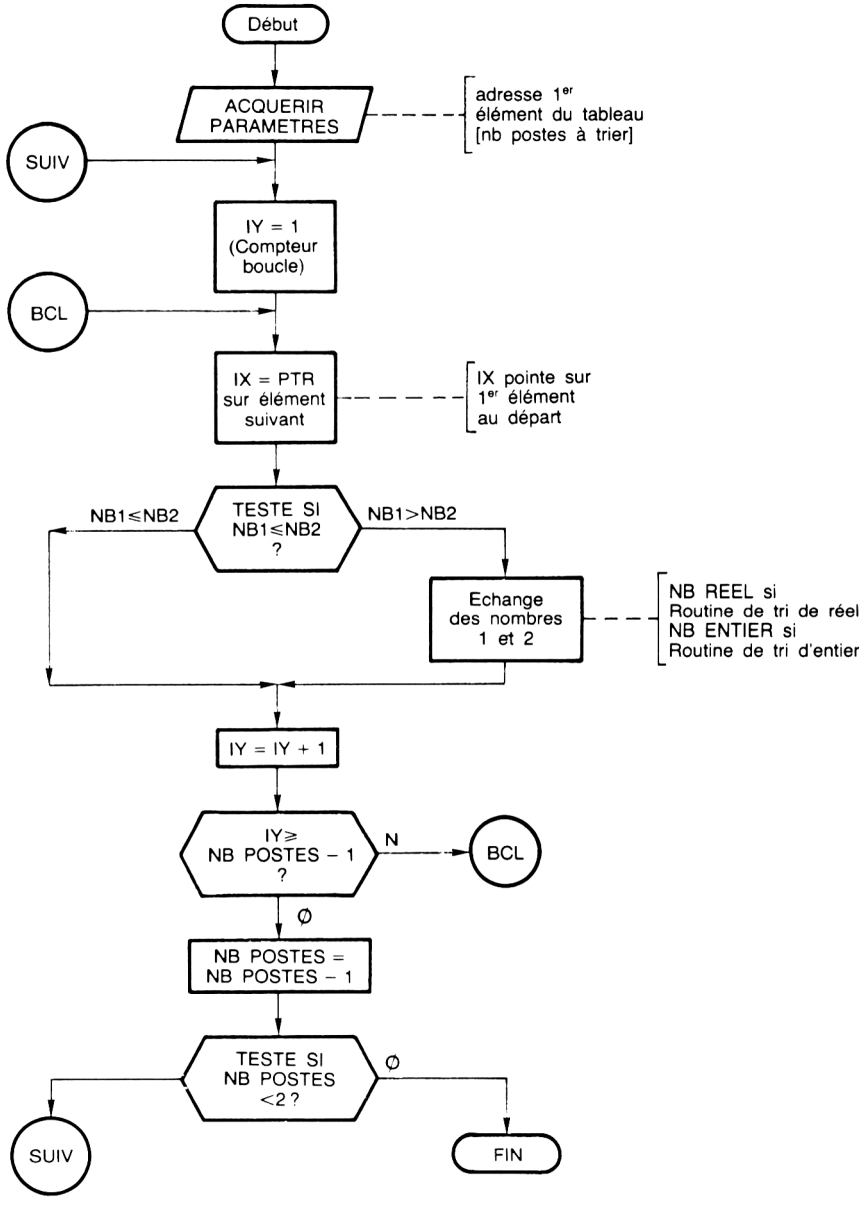
ROUTINE DE TRI DE TABLEAU DE RÉELS

La routine de tri de tableau de réels est similaire à la routine de tri de tableau d'entiers, à la différence qu'ici il s'agit de valeurs flottantes (soit sur 5 octets).
Il est à noter que la comparaison est effectuée par une routine située en RAM.
La version présentée tourne pour le CPC 6128 ; aussi pour la faire tourner sur le CPC 464, il vous faudra changer ce vecteur, en le remplaçant par #BD6A dans l'instruction source N° 390.

Mesure du temps d'exécution des routines de tri présentées

Pour vous montrer la puissance de tri assembleur, même quand il s'agit d'un « TRI BULLE », nous vous présentons un graphe des temps constatés en fonction du nombre de postes (ou articles) à trier en ordre croissant (voir GRAPHE DE TRI en fin de chapitre). Le test a été réalisé pour la routine de tri de tableau de chaînes ; nous avons constaté que, quelle que soit la taille de l'article à trier (1 à 255), et pour un même nombre d'articles, le temps de tri était irrémédiablement constant.

Routines de tri de tableau de nombres (réels et entiers)



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                  10 ;=* ROUTINE TRI DE TABLEAU D'ENTIER *=
6FA0              20         ent  #6fa0
6FA0              30         org  #6fa0
                  40 ;--* TRAITEMENT RSX *--
BCD1              50 rsx:    equ  #bcd1
6FA0  01AA6F      60         ld   bc,ext
6FA3  21D36F      70         ld   hl,space
6FA6  CDD1BC      80         call rsx
6FA9  C9          90         ret
6FAA  CE6F       100 ext:    defw nom
                 110 ;--* RECUPERATION PARAMETRES *--
                 120 ;--* HL=deb tabl / BC=nb elem a trier *--
6FAC  FE01       130         cp   1
6FAE  2805       140         jr   z,un
6FB0  FE02       150         cp   2
6FB2  280C       160         jr   z,deux
6FB4  C9         170         ret
                 180 ;--* TRAIT 1 PARAMETRE *--
6FB5  010000     190 un:     ld   bc,0
6FB8  DD6601     200         ld   h,(ix+1)
6FBB  DD6E00     210         ld   l,(ix+0)
6FBE  1840       220         jr   rout
                 230 ;--* TRAIT 2 PARAMETRES *--
6FC0  DD6603     240 deux:   ld   h,(ix+3)
6FC3  DD6E02     250         ld   l,(ix+2)
6FC6  DD4601     260         ld   b,(ix+1)
6FC9  DD4E00     270         ld   c,(ix+0)
6FCC  1832       280         jr   rout
                 290 ;
6FCE  545249     300 nom:    defm "TRI"
6FD1  C5         310         defb "E"+#80
6FD2  00         320         defb 0
6FD3             330 space:  defs 4
                 340 ;
                 350 ;--* DEBUT ROUTINE 'TRIE' *--
7000             360         org  #7000
B900             370 romhau: equ  #b900
C37D             380 affich: equ  #c37d
7000  CD00B9     390 rout:   call romhau
                 400 ;--* PTR HL SUR NB POSTES *--
7003  2B         410         dec  hl
7004  2B         420         dec  hl
                 430 ;--* SAUVE PTR HL DANS IX *--
7005  E5         440         push hl
```

```
7006  DDE1       450          pop  ix
                 460 ;--* TEST SI NB POSTES NUL *--
7008  0B         470          dec  bc
7009  CB78       480          bit  7,b
700B  2808       490          jr   z,post
                 500 ;--* SI NUL TRIER TOUS *--
700D  DD4601     510          ld   b,(ix+1)
7010  DD4E00     520          ld   c,(ix+0)
7013  181A       530          jr   postok
                 540 ;--* RECTIF NB POSTES *--
7015  03         550 post:    inc  bc
7016  CB78       560          bit  7,b
                 570 ;--* SI NEGATIF ARRET *--
7018  2815       580          jr   z,postok
701A  21F770     590          ld   hl,mess3
701D  CD7DC3     600          call affich
7020  C9         610          ret
7021  21A870     620 merr1:   ld   hl,mess1
7024  CD7DC3     630          call affich
7027  C9         640          ret
7028  21CF70     650 merr2:   ld   hl,mess2
702B  CD7DC3     660          call affich
702E  C9         670          ret
                 680 ;--* PTR HL SUR NB INDICES *--
702F  2B         690 postok: dec  hl
                 700 ;--* TEST SI PLUS D'UN INDICE *--
7030  3E01       710          ld   a,1
7032  BE         720          cp   (hl)
                 730 ;--* SI ERREUR ARRET *--
7033  20EC       740          jr   nz,merr1
                 750 ;--* PTR HL SUR TYPE VAR *--
7035  2B         760          dec  hl
7036  2B         770          dec  hl
7037  2B         780          dec  hl
                 790 ;--* TEST SI TYPE VAR = ENTIER *--
7038  3E01       800          ld   a,1
703A  BE         810          cp   (hl)
                 820 ;--* SI NON CHAINE ARRET *--
703B  20EB       830          jr   nz,merr2
                 840 ;--* SAUVE PTR IX ET NB POSTES *--
703D  DD22A670   850          ld   (ptrorg),ix
7041  ED43A470   860          ld   (nbpost),bc
                 870 ;--* CPT K = 1 *--
7045  FD210100   880 suiv:    ld   iy,1
                 890 ;--* PTR IX SUR PROCH. ELEM. *--
7049  DD23       900 bcl:     inc  ix
704B  DD23       910          inc  ix
                 920 ;--* ENTIER 1 DANS HL *--
704D  DD6601     930          ld   h,(ix+1)
```

```
7050  DD6E00     940          ld   l,(ix+0)
                 950 ;--* ENTIER 2 DANS DE *--
7053  DD5603     960          ld   d,(ix+3)
7056  DD5E02     970          ld   e,(ix+2)
                 980 ;--* SAUVE ENTIER 1 ET 2 *--
7059  E5         990          push hl
705A  D5        1000          push de
705B  A7        1010          and  a
705C  ED52      1020          sbc  hl,de
705E  CB7C      1030          bit  7,h
                1040 ;--* ALLER A E1LTE2 SI E1 INF A E2 *--
7060  201C      1050          jr   nz,e1lte2
                1060 ;--* RECUP ENTIER 1 ET 2 *--
7062  D1        1070          pop  de
7063  E1        1080          pop  hl
                1090 ;--* ECHANGE ENTIER 1 ET 2 *--
7064  DD7201    1100          ld   (ix+1),d
7067  DD7300    1110          ld   (ix+0),e
706A  DD7403    1120          ld   (ix+3),h
706D  DD7502    1130          ld   (ix+2),l
7070  180E      1140          jr   fintst
                1150 ;--* RECUP K ET VA A BCL *--
7072  FDE1      1160 pbcl:    pop  iy
7074  18D3      1170          jr   bcl
                1180 ;--* RECUP K ET VA A SUIV *--
7076  FDE1      1190 psuiv:   pop  iy
7078  DD2AA670  1200          ld   ix,(ptrorg)
707C  18C7      1210          jr   suiv
                1220 ;--* ENT 1 INF A ENT 2 *--
707E  D1        1230 e1lte2: pop  de
707F  E1        1240          pop  hl
                1250 ;--* RECUP K DANS DE *--
7080  FDE5      1260 fintst: push iy
7082  D1        1270          pop  de
                1280 ;--* K = K + 1 *--
7083  13        1290          inc  de
                1300 ;--* SAUVE K *--
7084  D5        1310          push de
                1320 ;--* RECUP NB POSTES (J) *--
7085  2AA470    1330          ld   hl,(nbpost)
                1340 ;--* J = J - 1 *--
7088  2B        1350          dec  hl
7089  A7        1360          and  a
708A  ED52      1370          sbc  hl,de
708C  CB7C      1380          bit  7,h
                1390 ;--* ALLER A PBCL SI K SUP OU EGAL A J-1 *--
708E  28E2      1400          jr   z,pbcl
                1410 ;--* SINON J=J-1 *--
```

```
7090  2AA470    1420          ld   hl,(nbpost)
7093  2B        1430          dec  hl
7094  22A470    1440          ld   (nbpost),hl
7097  110200    1450          ld   de,2
709A  A7        1460          and  a
709B  ED52      1470          sbc  hl,de
709D  CB7C      1480          bit  7,h
                1490 ;--* SI J SUP OU EGAL A 2 ALLER A PSUIV *--
709F  28D5      1500          jr   z,psuiv
                1510 ;--* REEQUILIBRE PILE *--
70A1  FDE1      1520          pop  iy
                1530 ;--* FIN SI J INFERIEUR A 2 *--
70A3  C9        1540          ret
                1550 ;
                1560 ;--* ZONES UTILISEES POUR DONNEES *--
70A4            1570 nbpost: defs 2                    ;NB POSTES TABL
70A6            1580 ptrorg: defs 2                    ;PTR 1ER ELEM
                1590 ;
                1600 ;--* MESSAGE ERREUR AVANT ARRET *--
70A8  6C652074  1610 mess1:  defm "le tableau a plus d'un indice / arret"
70CD  0A00      1620 saut1:  defb 10,0
70CF  6C652074  1630 mess2:  defm "le tableau n'est pas en 'nom%' / arret"
70F5  0A00      1640 saut2:  defb 10,0
70F7  6E622065  1650 mess3:  defm "nb elements a trier incorrect / arret"
711C  0A00      1660 saut3:  defb 10,0


Pass 2 errors: 00


affich C37D  bcl    7049  deux   6FC0
ellte2 707E  ext    6FAA  fintst 7080
merr1  7021  merr2  7028  mess1  70A8
mess2  70CF  mess3  70F7  nbpost 70A4
nom    6FCE  pbcl   7072  post   7015
postok 702F  psuiv  7076  ptrorg 70A6
romhau B900  rout   7000  rsx    BCD1
saut1  70CD  saut2  70F5  saut3  711C
space  6FD3  suiv   7045  un     6FB5


Table used:   328  from   900
Executes: 28576
```

Démo de tri d'un tableau d'entiers T% (20)

```
10 MODE 2
20 DIM t%(20)
30 CLS
40 FOR i=0 TO 20
50 a=RND(i)*1000:a=INT(a)
60 PRINT a
70 t%(i)=a
80 NEXT
90 LOCATE 25,25:PRINT"APPUYEZ SUR UNE TO
UCHE POUR CONTINUER":CALL &BB06
100 PRINT
110 a%=àt%(0)
120 |TRIE,a%
130 PRINT
140 FOR i=0 TO 20
150 PRINT t%(i)
160 NEXT
170 PRINT
180 END
```

Démo de tri d'une zone d'entiers

```
10 MODE 2
20 DATA 1,0,0,1,6,0,350,23,15,45,50,1
30 FOR i=0 TO 5
40 READ v:POKE 300+i,v
50 NEXT
60 FOR i=0 TO 5
70 READ v%:v$=HEX$(v%,4)
80 v1$="&"+LEFT$(v$,2):v2$="&"+RIGHT$(v$
,2)
90 v1=VAL(v1$):v2=VAL(v2$)
100 POKE 307+2*i,v1:POKE 306+2*i,v2
110 NEXT
120 PRINT"fin chargement"
130 PRINT
140 FOR i=0 TO 5
150 v2=PEEK(306+2*i):v1=PEEK(307+2*i)
160 v=v1*256+v2:PRINT v
165 NEXT
170 |TRIE,306
175 PRINT"apres tri"
180 FOR i=0 TO 5
190 v2=PEEK(306+2*i):v1=PEEK(307+2*i)
200 v=v1*256+v2:PRINT v
210 NEXT
```

```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                  10 ;=* ROUTINE TRI DE TABLEAU DE REEL *=
6FA0              20         ent  #6fa0
6FA0              30         org  #6fa0
                  40 ;--* TRAITEMENT RSX *--
BCD1              50 rsx:    equ  #bcd1
6FA0  01AA6F      60         ld   bc,ext
6FA3  21D36F      70         ld   hl,space
6FA6  CDD1BC      80         call rsx
6FA9  C9          90         ret
6FAA  CE6F       100 ext:    defw nom
                 110 ;--* RECUPERATION PARAMETRES *--
                 120 ;--* HL=deb tabl / BC=nb elem a trier *--
6FAC  FE01       130         cp   1
6FAE  2805       140         jr   z,un
6FB0  FE02       150         cp   2
6FB2  280C       160         jr   z,deux
6FB4  C9         170         ret
                 180 ;--* TRAIT 1 PARAMETRE *--
6FB5  010000     190 un:     ld   bc,0
6FB8  DD6601     200         ld   h,(ix+1)
6FBB  DD6E00     210         ld   l,(ix+0)
6FBE  1840       220         jr   rout
                 230 ;--* TRAIT 2 PARAMETRES *--
6FC0  DD6603     240 deux:   ld   h,(ix+3)
6FC3  DD6E02     250         ld   l,(ix+2)
6FC6  DD4601     260         ld   b,(ix+1)
6FC9  DD4E00     270         ld   c,(ix+0)
6FCC  1832       280         jr   rout
                 290 ;
6FCE  545249     300 nom:    defm "TRI"
6FD1  D2         310         defb "R"+#80
6FD2  00         320         defb 0
6FD3             330 space:  defs 4
                 340 ;
                 350 ;--* DEBUT ROUTINE 'TRIR' *--
7000             360         org  #7000
B900             370 romhau: equ  #b900
C37D             380 affich: equ  #c37d
BD8E             390 compar: equ  #bd8e                ;POUR CPC 6128
7000  CD00B9     400 rout:   call romhau
                 410 ;--* PTR HL SUR NB POSTES *--
7003  2B         420         dec  hl
7004  2B         430         dec  hl
                 440 ;--* SAUVE PTR HL DANS IX *--
```

```
7005  E5          450          push hl
7006  DDE1        460          pop  ix
                  470 ;--* TEST SI NB POSTES NUL *--
7008  0B          480          dec  bc
7009  CB78        490          bit  7,b
700B  2808        500          jr   z,post
                  510 ;--* SI NUL TRIER TOUS *--
700D  DD4601      520          ld   b,(ix+1)
7010  DD4E00      530          ld   c,(ix+0)
7013  181A        540          jr   postok
                  550 ;--* RECTIF NB POSTES *--
7015  03          560 post:    inc  bc
7016  CB78        570          bit  7,b
                  580 ;--* SI NEGATIF ARRET *--
7018  2815        590          jr   z,postok
701A  212C71      600          ld   hl,mess3
701D  CD7DC3      610          call affich
7020  C9          620          ret
7021  21E070      630 merr1:   ld   hl,mess1
7024  CD7DC3      640          call affich
7027  C9          650          ret
7028  210771      660 merr2:   ld   hl,mess2
702B  CD7DC3      670          call affich
702E  C9          680          ret
                  690 ;--* PTR HL SUR NB INDICES *--
702F  2B          700 postok: dec  hl
                  710 ;--* TEST SI PLUS D'UN INDICE *--
7030  3E01        720          ld   a,1
7032  BE          730          cp   (hl)
                  740 ;--* SI ERREUR ARRET *--
7033  20EC        750          jr   nz,merr1
                  760 ;--* PTR HL SUR TYPE VAR *--
7035  2B          770          dec  hl
7036  2B          780          dec  hl
7037  2B          790          dec  hl
                  800 ;--* TEST SI TYPE VAR = REEL *--
7038  3E04        810          ld   a,4
703A  BE          820          cp   (hl)
                  830 ;--* SI NON REEL ARRET *--
703B  20EB        840          jr   nz,merr2
                  850 ;--* PTR IX SUR TAILLE *--
703D  DD2B        860          dec  ix
703F  DD2B        870          dec  ix
7041  DD2B        880          dec  ix
                  890 ;--* SAUVE PTR IX ET NB POSTES *--
7043  DD22D470    900          ld   (ptrorg),ix
7047  ED43D270    910          ld   (nbpost),bc
                  920 ;--* CPT K = 1 *--
```

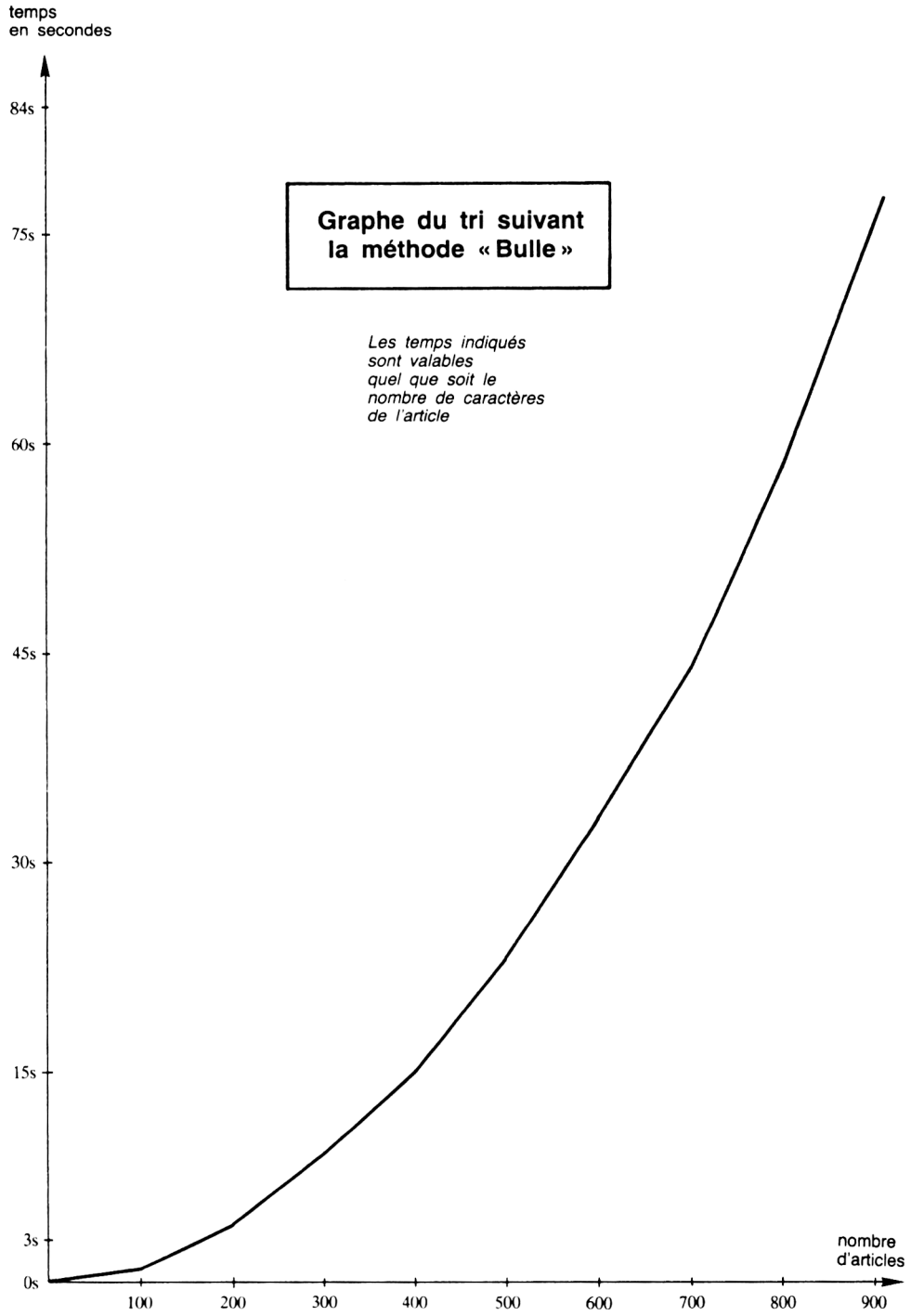
```
704B  FD210100   930 suiv:   ld   iy,1
                 940 ;--* PTR IX SUR PROCH. ELEM. *--
704F  DD23       950 bcl:    inc  ix
7051  DD23       960         inc  ix
7053  DD23       970         inc  ix
7055  DD23       980         inc  ix
7057  DD23       990         inc  ix
                1000 ;--* SAUVE IY ET IX *--
7059  FDE5      1010         push iy
705B  DDE5      1020         push ix
                1030 ;--* REEL 1 DANS REEL1 *--
705D  DDE5      1040         push ix
705F  E1        1050         pop  hl
7060  11D670    1060         ld   de,reel1
7063  010500    1070         ld   bc,5
7066  EDB0      1080         ldir
                1090 ;--* REEL 2 DANS REEL2 *--
7068  DDE5      1100         push ix
706A  E1        1110         pop  hl
706B  23        1120         inc  hl
706C  23        1130         inc  hl
706D  23        1140         inc  hl
706E  23        1150         inc  hl
706F  23        1160         inc  hl
7070  11DB70    1170         ld   de,reel2
7073  010500    1180         ld   bc,5
7076  EDB0      1190         ldir
                1200 ;--* COMPARE REEL 1 ET REEL 2 *--
7078  21D670    1210         ld   hl,reel1
707B  11DB70    1220         ld   de,reel2
707E  CD8EBD    1230         call compar
7081  FEFF      1240         cp   255
                1250 ;--* ALLER A FINTST SI R1 INF A R2 *--
7083  2829      1260         jr   z,fintst
                1270 ;--* ECHANGE REEL 1 ET 2 *--
7085  21DB70    1280         ld   hl,reel2
                1290 ;--* RECUP ET SAUVE IX *--
7088  D1        1300         pop  de
7089  D5        1310         push de
708A  010500    1320         ld   bc,5
                1330 ;--* TRANSFERT REEL 2 DANS ELEM 1 *--
708D  EDB0      1340         ldir
708F  21D670    1350         ld   hl,reel1
                1360 ;--* RECUP ET SAUVE IX *--
7092  D1        1370         pop  de
7093  D5        1380         push de
                1390 ;--* DE POINTE SUR ELEM 2 *--
7094  13        1400         inc  de
7095  13        1410         inc  de
```

```
7096  13        1420          inc  de
7097  13        1430          inc  de
7098  13        1440          inc  de
7099  010500    1450          ld   bc,5
                1460 ;--* TRANSFERT REEL 1 DANS ELEM 2 *--
709C  EDB0      1470          ldir
                1480 ;--* RECUP ET SAUVE IX *--
709E  D1        1490          pop  de
709F  D5        1500          push de
70A0  180C      1510          jr   fintst
                1520 ;--* RECUP K ET VA A BCL *--
70A2  FDE1      1530 pbcl:    pop  iy
70A4  18A9      1540          jr   bcl
                1550 ;--* RECUP K ET VA A SUIV *--
70A6  FDE1      1560 psuiv:   pop  iy
70A8  DD2AD470  1570          ld   ix,(ptrorg)
70AC  189D      1580          jr   suiv
                1590 ;--* RECUP IX PUIS K DANS DE *--
70AE  DDE1      1600 fintst: pop  ix
70B0  D1        1610          pop  de
                1620 ;--* K = K + 1 *--
70B1  13        1630          inc  de
                1640 ;--* SAUVE K *--
70B2  D5        1650          push de
                1660 ;--* RECUP NB POSTES (J) *--
70B3  2AD270    1670          ld   hl,(nbpost)
                1680 ;--* J = J - 1 *--
70B6  2B        1690          dec  hl
70B7  A7        1700          and  a
70B8  ED52      1710          sbc  hl,de
70BA  CB7C      1720          bit  7,h
                1730 ;--* ALLER A PBCL SI K SUP OU EGAL A J-1 *--
70BC  28E4      1740          jr   z,pbcl
                1750 ;--* SINON J=J-1 *--
70BE  2AD270    1760          ld   hl,(nbpost)
70C1  2B        1770          dec  hl
70C2  22D270    1780          ld   (nbpost),hl
70C5  110200    1790          ld   de,2
70C8  A7        1800          and  a
70C9  ED52      1810          sbc  hl,de
70CB  CB7C      1820          bit  7,h
                1830 ;--* SI J SUP OU EGAL A 2 ALLER A PSUIV *--
70CD  28D7      1840          jr   z,psuiv
                1850 ;--* REEQUILIBRE PILE *--
70CF  FDE1      1860          pop  iy
                1870 ;--* FIN SI J INFERIEUR A 2 *--
70D1  C9        1880          ret
                1890 ;
                1900 ;--* ZONES UTILISEES POUR DONNEES *--
```

```
70D2              1910 nbpost: defs 2                    ;NB POSTES TABL
70D4              1920 ptrorg: defs 2                    ;PTR 1ER ELEM
70D6              1930 reel1:  defs 5                    ;FLOTTANT 1
70DB              1940 reel2:  defs 5                    ;FLOTTANT 2
                  1950 ;
                  1960 ;--* MESSAGE ERREUR AVANT ARRET *--
70E0  6C652074    1970 mess1:  defm "le tableau a plus d'un indice / arret"
7105  0A00        1980 saut1:  defb 10,0
7107  6C652074    1990 mess2:  defm "le tableau n'est pas en 'nom' / arret"
712C  6E622065    2000 mess3:  defm "nb elements a trier incorrect / arret"
7151  0A00        2010 saut3:  defb 10,0


Pass 2 errors: 00


affich C37D  bcl    704F  compar BD8E
deux   6FC0  ext    6FAA  fintst 70AE
merr1  7021  merr2  7028  mess1  70E0
mess2  7107  mess3  712C  nbpost 70D2
nom    6FCE  pbcl   70A2  post   7015
postok 702F  psuiv  70A6  ptrorg 70D4
reel1  70D6  reel2  70DB  romhau B900
rout   7000  rsx    BCD1  saut1  7105
saut3  7151  space  6FD3  suiv   704B
un     6FB5


Table used:   340  from   900
Executes: 28576
```

Démo de tri d'un tableau de réels T (20)

```
10 MODE 2
20 DIM t(20)
30 CLS
40 FOR i=0 TO 20
50 a=RND(i)*1000
60 PRINT a
70 t(i)=a
80 NEXT
90 LOCATE 25,25:PRINT"APPUYEZ SUR UNE TO
UCHE POUR CONTINUER":CALL &BB06
100 PRINT
110 a%=àt(0)
120 |TRIR,a%
130 PRINT
140 FOR i=0 TO 20
150 PRINT t(i)
160 NEXT
170 PRINT
180 END
```


temps
en secondes
84s
75s
60s
45s
30s
15s
3s
0s
Graphe du tri suivant
la méthode « Bulle »
Les temps indiqués
sont valables
quel que soit le
nombre de caractères
de l'article
nombre
d'articles
100
200
300
400
500
600
700
800
900

CHAPITRE 4

GRAPHISMES

GÉNÉRALITÉS

L'Amstrad possède d'excellentes capacités graphiques dont il est important de savoir tirer parti afin d'améliorer la qualité de ses réalisations. C'est l'objet de ce chapitre. Celui-ci sera partagé en trois parties :

(1) extensions BASIC pour le CPC 464,
(2) traçage de figures géométriques,
(3) traitement sur la mémoire écran.

Pour chaque routine étudiée, vous trouverez : sa fonction, un organigramme fonctionnel, la liste assembleur, des explications concernant la structure et le contenu du programme, et parfois un exemple d'utilisation qui pourra être accompagné d'une copie de page écran.

Les programmes suivants utilisent au maximum les routines système résidant dans la RAM des CPC. Celles-ci seront donc décrites avant les listes assembleur. Il sera précisé pour chacune d'entre elles leurs adresses respectives dans la RAM du CPC 6128 et dans celle du CPC 464 car il arrive parfois, notamment pour les routines de calculs mathématiques, que leurs emplacements soient différents selon la machine utilisée.

En règle générale, les routines proposées seront exploitables de deux manières différentes :

(1) comme extensions BASIC précédées du symbole | et suivies de la liste des paramètres nécessaires à leur bon fonctionnement.

(2) comme routines assembleur pouvant être appelées dans un programme BASIC par l'ordre CALL, suivi — si besoin est — des paramètres, ou appelables directement dans un programme assembleur, à condition que les données soient bien fournies pour permettre une exécution correcte.

EXTENTIONS BASIC POUR CPC 464

COULEURS GRAPHIQUES

Fonction : Choix de la couleur de l'encre du stylo et du papier pour des tracés graphiques.

Contrairement au CPC 6128, le CPC 464 ne possède pas l'ordre BASIC permettant d'effectuer le choix des couleurs des encres du stylo et du papier dans le cas de tracés graphiques.

Une solution consiste à faire suivre l'ordre d'un paramètre supplémentaire indiquant uniquement la couleur de l'encre du tracé en cours.

Une autre solution est envisageable mais très coûteuse : il s'agit d'utiliser les codes de contrôle, soit CHR$(N), avec N inférieur à 32. Le RSX suivant vous permettra de résoudre facilement ce problème à moindre frais.

Syntaxe : |COULGRAF,N1,[N2]

avec N1 : numéro de l'encre du stylo.
N2 : numéro de l'encre du papier.

Comme la couleur du fond ne peut apparaître que par l'ordre CLG, effaçage de la fenêtre graphique, il est préférable de permettre à l'utilisateur l'envoi facultatif du deuxième paramètre.

Routines système utilisées

#BCD1 : permet la création d'une extension.
#BBDE : encre du stylo graphique.
#BBE4 : encre du papier graphique.

Ces adresses restent inchangées quel que soit le CPC utilisé.

Explications

La routine commence par tester si l'utilisateur a fait suivre le RSX de plus de 2 paramètres. Dans ce cas, un retour au BASIC a immédiatement lieu sans affichage de message d'erreur.

Le programme cherche ensuite s'il y a un ou deux paramètres :

Cas d'utilisation d'un paramètre :

Pour un seul paramètre s'effectue alors le positionnement de l'encre du stylo. Celle-ci se trouvant à l'adresse pointée par le registre IX sera chargée dans le registre C avant l'appel de la routine #BBDE.

Cas d'utilisation de deux paramètres :

Pour deux paramètres, la routine positionne d'abord le numéro de l'encre du papier (#BBE4), puis celle du stylo comme précédemment.

Un exemple d'utilisation de cette extension sera fourni avec la deuxième routine concernant les rectangles dans le paragraphe 2.

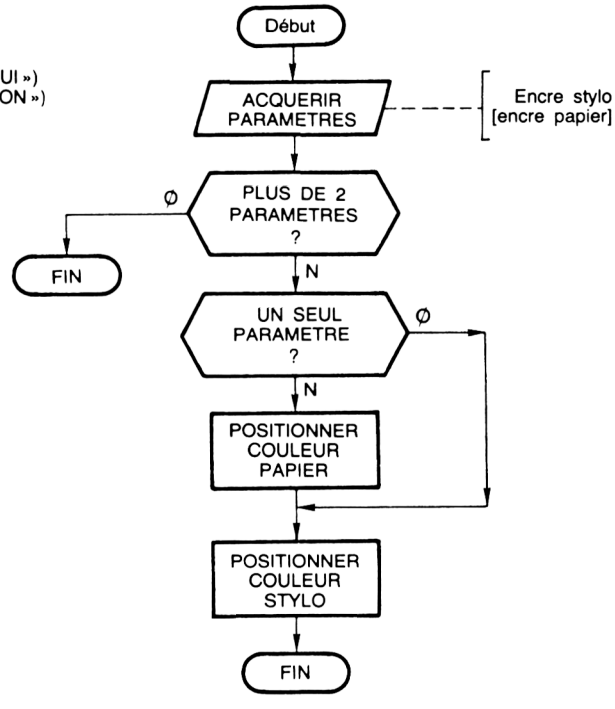
Il est à noter que les paramètres, permettant une bonne exécution de ce RSX, se rapportent au numéro des encres et non au numéro des couleurs.

Il conviendra donc de bien définir auparavant les couleurs attribuées à chaque encre...

Routine de positionnement des couleurs graphiques (CPC 464)

Légende :

0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → lettre (pour « NON »)



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                  10 ;=*  COULEURS GRAPHIQUES CPC 464  *=
                  20 ;
6FA0              30         ent  #6fa0
6FA0              40         org  #6fa0
                  50 ;
BCD1              60 rsx:    equ  #bcd1
BBDE              70 pen:    equ  #bbde
BBE4              80 paper:  equ  #bbe4
                  90 ;
6FA0  01AA6F     100         ld   bc,ext
6FA3  21D06F     110         ld   hl,space
6FA6  CDD1BC     120         call rsx
6FA9  C9         130         ret
6FAA  C76F       140 ext:    defw nom
                 150 ;--* TEST SI 2 PARAMETRES *--
6FAC  FE02       160         cp   2
6FAE  280B       170         jr   z,papier
                 180 ;--* TEST SI 1 SEUL PARAMETRE *--
6FB0  FE01       190         cp   1
6FB2  C0         200         ret  nz
                 210 ;--* ENCRE DU STYLO *--
6FB3  DD4E00     220         ld   c,(ix+0)
6FB6  79         230 stylo:  ld   a,c
6FB7  CDDEBB     240         call pen
6FBA  C9         250         ret
                 260 ;--* ENCRE DU PAPIER *--
6FBB  DD5E00     270 papier: ld   e,(ix+0)
6FBE  7B         280         ld   a,e
6FBF  CDE4BB     290         call paper
6FC2  DD4E02     300         ld   c,(ix+2)
6FC5  18EF       310         jr   stylo
                 320 ;
6FC7  434F554C   330 nom:    defm "COULGRA"
6FCE  C6         340         defb "F"+#80
6FCF  00         350         defb 0
6FD0             360 space:  defs 4
```

Pass 2 errors: 00

```
ext    6FAA  nom    6FC7  paper  BBE4
papier 6FBB  pen    BBDE  rsx    BCD1
space  6FD0  stylo  6FB6
```

Table used: 102 from 900
Executes: 28576

FENÊTRE GRAPHIQUE

Fonction : Création d'une fenêtre graphique.

Il s'agit ici de délimiter une fenêtre dans laquelle viendront s'exécuter les ordres graphiques comme le choix des couleurs de l'encre du stylo et du papier, ainsi que tous les ordres de traçage graphique.

L'intérêt est de concevoir ainsi des écrans contenant à la fois du texte et du graphisme, sans interférence de l'un sur l'autre. Il sera alors possible d'effacer le contenu de la fenêtre graphique sans effacer le texte, ou de changer l'encre d'écriture des caractères sans changer celle du graphisme.

Syntaxe : |FENGRAF,X1,X2,Y1,Y2

avec X1 : coordonnée du bord gauche de la fenêtre.
X2 : coordonnée du bord droit de la fenêtre.
Y1 : coordonnée du bord haut de la fenêtre.
Y2 : coordonnée du bord bas de la fenêtre.

X1 et X2 compris entre 0 et 639.
Y1 et Y2 compris entre 0 et 399.

Routines système utilisées

#BBCF : placement des bords verticaux de la fenêtre.
#BBD2 : placement des bords horizontaux de la fenêtre.

Explications

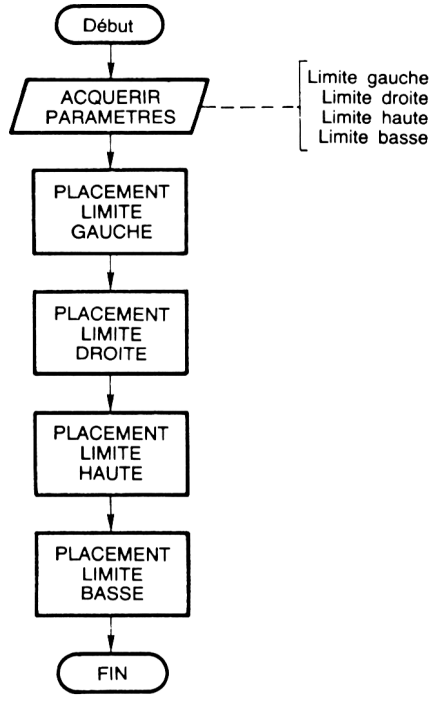
La routine est fort simple. Après lecture des paramètres, les registres DE et HL contiennent X1 et X2 et la routine #BBCF est appelée.

Les registres DE et HL sont ensuite chargés avec Y1 et Y2 avant l'appel de la routine #BBD2 qui fixe les bords verticaux.

Il est à noter que le système corrige l'envoi de paramètres trop grands. De même, le bord gauche est toujours déterminé par la plus petite des valeurs entre X1 et X2 ; l'ordre n'a donc que peu d'importance. Le même phénomène se reproduit avec les limites horizontales pour lesquelles le bord bas est fixé par la plus petite des valeurs entre Y1 ou Y2.

Cette routine de détermination de la fenêtre graphique peut être complétée par la routine précédente pour fixer les couleurs des encres du stylo et du papier...

Routine de positionnement d'une fenêtre graphique (CPC 464)



```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                 10 ;=*  FENETRES GRAPHIQUES CPC 464  *=
                 20 ;
6FA0             30          ent  #6fa0
6FA0             40          org  #6fa0
                 50 ;
                 60 ;--* TRAITEMENT RSX *--
                 70 ;
BCD1             80 rsx:     equ  #bcd1
BBCF             90 borver:  equ  #bbcf
BBD2            100 borhor:  equ  #bbd2
                110 ;
6FA0  01AA6F    120          ld   bc,ext
6FA3  21E16F    130          ld   hl,space
6FA6  CDD1BC    140          call rsx
6FA9  C9        150          ret
                160 ;
                170 ;--* RECUPERATION DES PARAMETRES *--
                180 ;
6FAA  D96F      190 ext:     defw nom
6FAC  DD4607    200          ld   b,(ix+7)
6FAF  DD4E06    210          ld   c,(ix+6)
6FB2  ED431570  220          ld   (gauche),bc
6FB6  DD4605    230          ld   b,(ix+5)
6FB9  DD4E04    240          ld   c,(ix+4)
6FBC  ED431770  250          ld   (droite),bc
6FC0  DD4603    260          ld   b,(ix+3)
6FC3  DD4E02    270          ld   c,(ix+2)
6FC6  ED431970  280          ld   (haut),bc
6FCA  DD4601    290          ld   b,(ix+1)
6FCD  DD4E00    300          ld   c,(ix+0)
6FD0  ED431B70  310          ld   (bas),bc
6FD4  FE04      320          cp   4
6FD6  C0        330          ret  nz
6FD7  1827      340          jr   rout
                350 ;
6FD9  46454E47  360 nom:     defm "FENGRA"
6FDF  C6        370          defb "F"+#80
6FE0  00        380          defb 0
6FE1            390 space:   defs 4
                400 ;
                410 ;--* TRAITEMENT CALL *--
                420 ;
7000            430          org  #7000
                440 ;
                450 ;--* PLACEMENT LIMITES VERTICALES *--
```

```
7000  ED5B1570   460 rout:   ld   de,(gauche)
7004  2A1770     470         ld   hl,(droite)
7007  CDCFBB     480         call borver
                 490 ;--* PLACEMENT LIMITES HORIZONTALES *--
700A  ED5B1970   500         ld   de,(haut)
700E  2A1B70     510         ld   hl,(bas)
7011  CDD2BB     520         call borhor
7014  C9         530         ret
                 540 ;
7015             550 gauche: defs 2                    ;STO BORD GAUCHE
7017             560 droite: defs 2                    ;STO BORD DROIT
7019             570 haut:   defs 2                    ;STO BORD HAUT
701B             580 bas:    defs 2                    ;STO BORD BAS

Pass 2 errors: 00

bas    701B  borhor BBD2  borver BBCF
droite 7017  ext    6FAA  gauche 7015
haut   7019  nom    6FD9  rout   7000
rsx    BCD1  space  6FE1

Table used:   139  from   900
Executes: 28576
```

TRACÉS DE FIGURES GÉOMÉTRIQUES

GÉNÉRALITÉS

Ce paragraphe vous propose d'étudier des routines permettant de tracer des figures géométriques telles que : triangle, rectangle, cercle, ellipse et polygone régulier à n côtés.

Bien entendu le nombre de ces routines aurait pû être étendu. On pourrait envisager le tracement de losanges, de parallélogrammes ou de triangles particuliers. Ce n'est pas notre propos que de proposer une liste interminable de figures et nous pensons qu'après l'étude des routines suivantes, votre imagination aidant, vous serez en mesure de créer vos propres applications...

Pour certains des programmes suivants, il est nécessaire d'effectuer des calculs mathématiques. Ils seront réalisés à partir des routines système disponibles dans la RAM des CPC.

Cependant un problème se pose. En effet, d'une part les adresses diffèrent suivant la machine utilisée, d'autre part le CPC 6128 ne dispose pas des routines permettant de traiter les calculs sur les nombres entiers tels que multiplication et division.

Par souci d'harmonisation et pour que toutes les routines proposées puissent tourner sur tous les modèles de CPC, nous avons choisi d'utiliser le traitement des calculs en virgule flottante alors que parfois, pour le 464, ces calculs auraient pû être effectués en nombres entiers. L'utilisateur pourra, s'il possède cette dernière machine, transformer les listes assembleur et faire les calculs à l'aide des vecteurs mathématiques propres à son appareil.

Les listes assembleur sont prévues pour le CPC 6128, mais dans la présentation des routines, vous trouverez, si nécessaire, les équivalences pour le CPC 464...

TRACÉ DE TRIANGLE

Fonction : Tracé d'un triangle quelconque.

Cette routine permet de tracer un triangle, dans la fenêtre graphique en cours si celle-ci a été définie, sinon directement sur tout l'écran. Pour cela, il suffira de faire suivre l'ordre RSX par les coordonnées des trois sommets du triangle ou fournir les paramètres nécessaires directement au programme assembleur.

Pour tracer des triangles particuliers, il faudra jouer sur les coordonnées des sommets.

Vous trouverez également dans cette routine une particularité commune à tous les autres programmes de ce chapitre, à savoir la sauvegarde de l'origine en cours. En effet, la routine utilisée pour effectuer le tracé d'un point à l'écran utilise, comme origine de référence, l'origine en cours définie par l'utilisateur et positionne le point par rapport à celle-ci. Il aurait donc fallu en tenir compte dans la détermination des coordonnées des sommets.

Pour éviter ces calculs, nous avons choisi d'exprimer toutes les coordonnées par rapport à une origine absolue (0,0), située en bas et à gauche de l'écran. Dans ce cas, il faut auparavant sauver l'origine en cours, ce qui est fait en début de programme ; puis positionner de la nouvelle origine en (0,0), et rétablir de l'origine initiale à l'issue de l'exécution de la routine.

Syntaxe : |TRIANGLE,X1,Y1,X2,Y2,X3,Y3

avec X1,Y1 : coordonnées du premier sommet.
X2,Y2 : coordonnées du deuxième sommet.
X3,Y3 : coordonnées du troisième sommet.

X1, X2 et X3 compris entre 0 et 639.
Y1, Y2 et Y3 compris entre 0 et 399.

Routines système utilisées

#BBCC : recherche de l'origine en cours.
#BBC9 : positionnement de la nouvelle origine.
#BBF6 : tracé d'une droite.
#BBEA : tracé d'un point.

Aucun changement quel que soit le CPC utilisé.

Explications

Après la réception des paramètres, fournis avec l'instruction RSX ou directement chargés à leur emplacement mémoire, la routine commence par sauver l'origine en cours dans les variables ORGX et ORGY. La nouvelle origine est alors fixée, de coordonnées 0,0.

Les registres DE et HL sont chargés avec les coordonnées du premier sommet, X1 et Y1. Celles-ci sont placées dans la pile afin de pouvoir être rappelées lors du tracement du dernier côté. Puis la routine trace le premier sommet par appel de #BBEA.

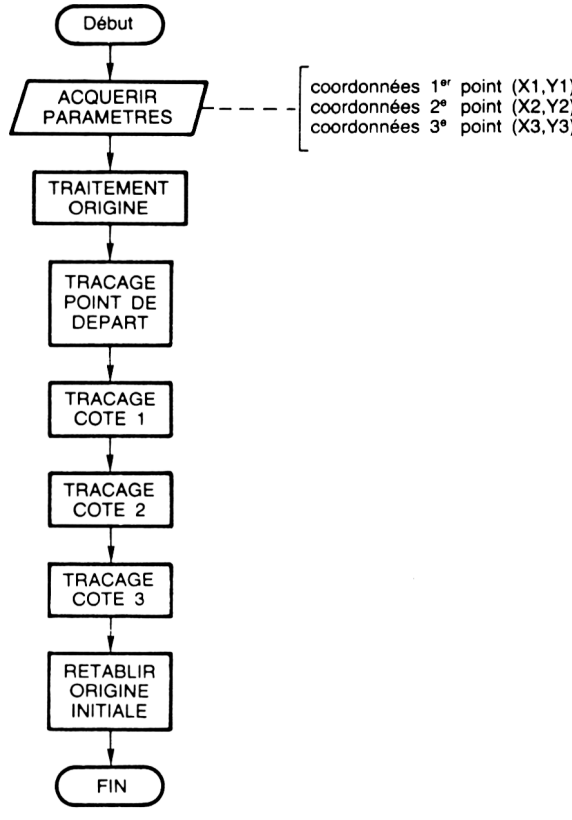
Les coordonnées du deuxième sommet, X2 et Y2, sont placées dans les registres DE et HL afin de tracer la droite reliant la dernière position du curseur graphique au point dont les coordonnées viennent d'être fournies. Ceci s'effectue grâce à la routine #BBF6. On agit de même pour le troisième sommet de coordonnées X3 et Y3. Pour tracer le dernier côté, il suffit de dépiler X1 et Y1 et d'appeler à nouveau la routine #BBF6.
Pour conclure, on replace l'origine initiale par appel de #BBC9 après chargement de ORGX et ORGY dans les registres DE et HL...

Démonstration

Le court programme BASIC suivant vous permettra de tracer un superbe dessin grâce à cette nouvelle instruction.

```
10 MODE2
20 FOR I=0 TO 50
30 |TRIANGLE,12×I,100−2×I,100+8×I,3×I,150−3×I,400−6×i
40 NEXT
```

Routine de tracé de triangle



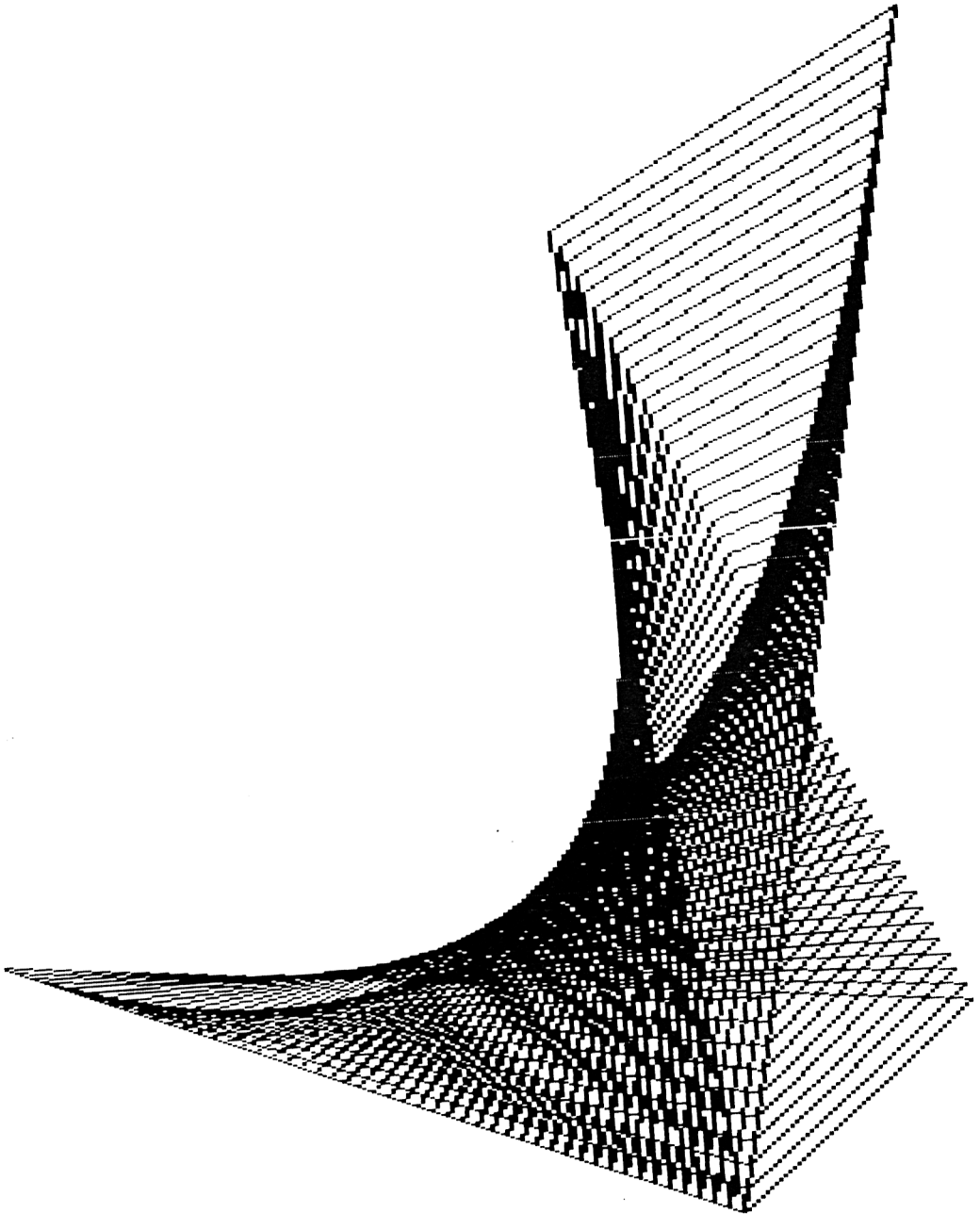
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                  10 ;=* ROUTINE TRIANGLE CPC 464 ET 6128 *=
                  20 ;
6FA0              30          ent  #6fa0
6FA0              40          org  #6fa0
                  50 ;
                  60 ;--* TRAITEMENT RSX *--
                  70 ;
BCD1              80 rsx:     equ  #bcd1
6FA0  01AA6F      90          ld   bc,ext
6FA3  21F46F     100          ld   hl,space
6FA6  CDD1BC     110          call rsx
6FA9  C9         120          ret
6FAA  EB6F       130 ext:     defw nom
                 140 ;--* RECUPERATION DES PARAMETRES *--
6FAC  DD4601     150          ld   b,(ix+1)
6FAF  DD4E00     160          ld   c,(ix+0)
6FB2  ED434D70   170          ld   (y3),bc
6FB6  DD5603     180          ld   d,(ix+3)
6FB9  DD5E02     190          ld   e,(ix+2)
6FBC  ED534B70   200          ld   (x3),de
6FC0  DD6605     210          ld   h,(ix+5)
6FC3  DD6E04     220          ld   l,(ix+4)
6FC6  224970     230          ld   (y2),hl
6FC9  DD4607     240          ld   b,(ix+7)
6FCC  DD4E06     250          ld   c,(ix+6)
6FCF  ED434770   260          ld   (x2),bc
6FD3  DD6609     270          ld   h,(ix+9)
6FD6  DD6E08     280          ld   l,(ix+8)
6FD9  224570     290          ld   (y1),hl
6FDC  DD460B     300          ld   b,(ix+11)
6FDF  DD4E0A     310          ld   c,(ix+10)
6FE2  ED434370   320          ld   (x1),bc
6FE6  FE06       330          cp   6
6FE8  C0         340          ret  nz
6FE9  1815       350          jr   rout
                 360 ;
6FEB  54524941   370 nom:     defm "TRIANGL"
6FF2  C5         380          defb "E"+#80
6FF3  00         390          defb 0
6FF4             400 space:   defs 4
                 410 ;
                 420 ;
                 430 ;--* TRAITEMENT CALL *--
                 440 ;
```

```
7000              450          org  #7000
                  460 ;
BBCC              470 rechor: equ  #bbcc
BBC9              480 origin: equ  #bbc9
BBF6              490 tracdr: equ  #bbf6
BBEA              500 tracpi: equ  #bbea
                  510 ;
7000  CDCCBB      520 rout:   call rechor
                  530 ;--* SAUVEGARDE DE L'ORIGINE *--
7003  ED534F70    540          ld   (orgx),de
7007  225170      550          ld   (orgy),hl
                  560 ;--* MISE DE L'ORIGINE A (0,0) *--
700A  110000      570          ld   de,0
700D  210000      580          ld   hl,0
7010  CDC9BB      590          call origin
                  600 ;--* TRACAGE POINT DEPART *--
7013  ED5B4370    610          ld   de,(x1)
7017  2A4570      620          ld   hl,(y1)
701A  D5          630          push de
701B  E5          640          push hl
701C  CDEABB      650          call tracpi
                  660 ;--* TRACAGE DES COTES *--
701F  ED5B4770    670          ld   de,(x2)
7023  2A4970      680          ld   hl,(y2)
7026  CDF6BB      690          call tracdr
7029  ED5B4B70    700          ld   de,(x3)
702D  2A4D70      710          ld   hl,(y3)
7030  CDF6BB      720          call tracdr
7033  E1          730          pop  hl
7034  D1          740          pop  de
7035  CDF6BB      750          call tracdr
                  760 ;--* REMISE ORIGINE INITIALE *--
7038  ED5B4F70    770          ld   de,(orgx)
703C  2A5170      780          ld   hl,(orgy)
703F  CDC9BB      790          call origin
7042  C9          800          ret
                  810 ;
7043              820 x1:      defs 2                    ;X 1er POINT
7045              830 y1:      defs 2                    ;Y 1er POINT
7047              840 x2:      defs 2                    ;X 2eme POINT
7049              850 y2:      defs 2                    ;Y 2eme POINT
704B              860 x3:      defs 2                    ;X 3eme POINT
704D              870 y3:      defs 2                    ;Y 3eme POINT
704F              880 orgx:    defs 2                    ;ORIG. INIT. X
7051              890 orgy:    defs 2                    ;ORIG. INIT. Y
```

```
Pass 2 errors: 00

ext    6FAA  nom    6FEB  orgx   704F
orgy   7051  origin BBC9  rechor BBCC
rout   7000  rsx    BCD1  space  6FF4
tracdr BBF6  tracpi BBEA  x1     7043
x2     7047  x3     704B  y1     7045
y2     7049  y3     704D

Table used:   194  from   900
Executes: 28576
```



TRACÉ DE RECTANGLE

Fonction : tracé d'un rectangle.

Le programme suivant permet de tracer des rectangles dont les bords sont parallèles aux côtés de l'écran. Pour cela, il faudra définir le point de départ de la figure, qui sera le coin bas à gauche, ainsi bien entendu que la longueur et la largeur.

A partir de là, on pourrait envisager de déterminer les quatre sommets et les relier par des droites comme cela était le cas pour le tracé des triangles. Nous avons préféré une solution plus rapide qui utilise la capacité des CPC à tracer des droites. La dernière position du curseur sera reliée à un point dont on indique soit les coordonnées absolues par rapport à l'origine, soit les coordonnées relatives par rapport à la dernière position du curseur.

Dans le cas du rectangle, il est préférable de choisir la deuxième possibilité car le déplacement à effectuer sera toujours de + ou − la longueur ou + ou − la largeur suivant le côté à tracer.

Syntaxe : |RECTANGLE,A,B,C,D

avec A = coordonnée X de départ (comprise entre 0 et 639).
B = coordonnée Y de départ (comprise entre 0 et 399).
C = côté horizontal.
D = côté vertical.

Le CPC ne tracera que les points inclus dans la fenêtre graphique si celle-ci est définie.

Routines système utilisées

#BBCC : recherche de l'origine en cours.
#BBC9 : positionnement de la nouvelle origine.
#BBEA : tracé d'un point.
#BBF6 : tracé d'une droite vers une position absolue.
#BBF9 : tracé d'une droite vers une position relative.
Ces adresses sont les mêmes pour tous les CPC.

Explications

Les paramètres sont récupérés dans les variables DEPX et DEPY pour les coordonnées du coin en bas à gauche, et dans COTH et COTV pour la longueur de chaque côté.

L'ancienne origine est sauvegardée dans ORGX et ORGY et la nouvelle positionnée en 0,0. La routine trace, par appel de #BBEA, le point de départ, de coordonnée DEPX et DEPY.

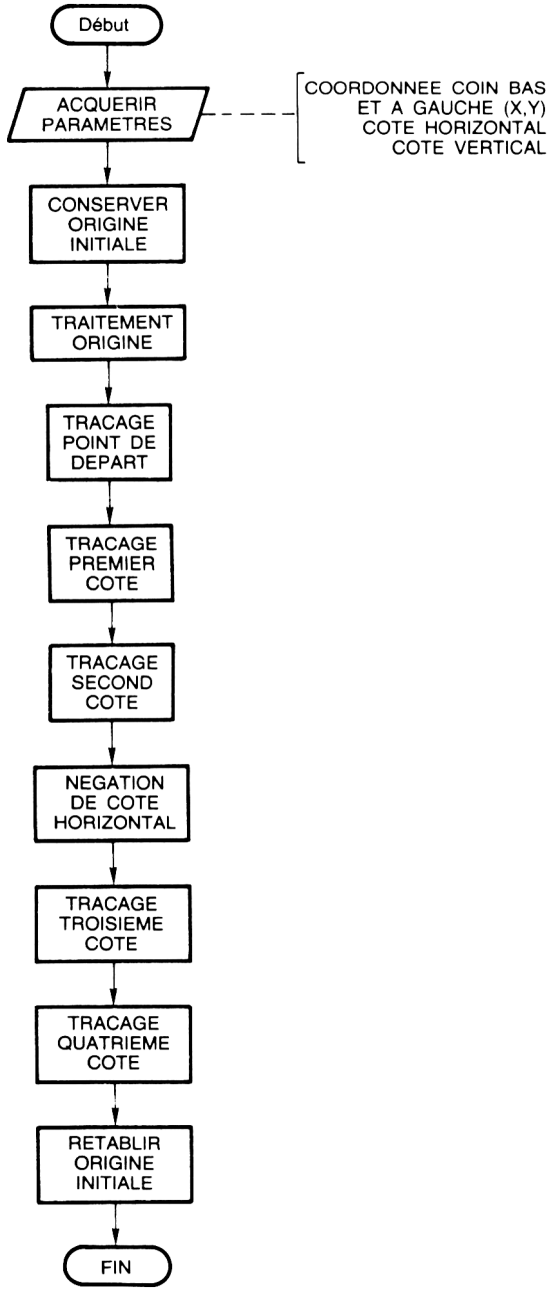
Le premier côté est alors tracé à l'aide de la routine #BBF9 qui relie la dernière position du curseur à un nouveau point défini par un déplacement relatif de COTH vers la droite. Le deuxième côté est tracé de la même manière, avec cette fois un déplacement vertical de COTV.

Le côté suivant correspond à un déplacement horizontal vers la gauche de l'écran, donc d'une valeur négative, plus précisément de — COTH. Il faut alors calculer cette valeur. Nous aurions pu faire appel à un vecteur mathématique pour effectuer ce travail. Nous avons préféré utiliser le complément à 2 qui est, en binaire, la négation d'un nombre. Pour cela, COTH est chargé dans HL, puis H et L transitent par le registre A où s'effectue le complément, avant que HL soit incrémenté de 1 pour obtenir — COTH. Le troisième côté est alors tracé, toujours grâce à la routine #BBF9.

Pour le dernier côté, il suffit de tracer la droite reliant la dernière position du curseur au point de départ, ce qui se fait par la routine #BBF6.

On conclut en replaçant l'origine initiale sauvegardée en début de programme.

Routine de tracé de rectangle non orientable



```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                  10 ;=* ROUTINE RECTANGLE CPC 464 ET 6128 *=
                  20 ;
6FA0              30          ent  #6fa0
6FA0              40          org  #6fa0
                  50 ;
                  60 ;--* TRAITEMENT RSX *--
                  70 ;
BCD1              80 rsx:     equ  #bcd1
6FA0  01AA6F      90          ld   bc,ext
6FA3  21E26F     100          ld   hl,space
6FA6  CDD1BC     110          call rsx
6FA9  C9         120          ret
6FAA  D86F       130 ext:     defw nom
                 140 ;--* RECUPERATION DES PARAMETRES *--
6FAC  DD4601     150          ld   b,(ix+1)
6FAF  DD4E00     160          ld   c,(ix+0)
6FB2  ED435870   170          ld   (cotv),bc
6FB6  DD5603     180          ld   d,(ix+3)
6FB9  DD5E02     190          ld   e,(ix+2)
6FBC  ED535670   200          ld   (coth),de
6FC0  DD6605     210          ld   h,(ix+5)
6FC3  DD6E04     220          ld   l,(ix+4)
6FC6  225C70     230          ld   (depy),hl
6FC9  DD4607     240          ld   b,(ix+7)
6FCC  DD4E06     250          ld   c,(ix+6)
6FCF  ED435A70   260          ld   (depx),bc
6FD3  FE04       270          cp   4
6FD5  C0         280          ret  nz
6FD6  1828       290          jr   rout
                 300 ;
6FD8  52454354   310 nom:     defm "RECTANGL"
6FE0  C5         320          defb "E"+#80
6FE1  00         330          defb 0
6FE2             340 space:   defs 4
                 350 ;
                 360 ;
                 370 ;--* TRAITEMENT CALL *--
                 380 ;
7000             390          org  #7000
                 400 ;
BBCC             410 rechor: equ  #bbcc
BBC9             420 origin: equ  #bbc9
BBEA             430 tracpi: equ  #bbea
BBF9             440 tracdl: equ  #bbf9
```

```
BBF6            450 tracd2: equ  #bbf6
                460 ;
7000  CDCCBB    470 rout:   call rechor
                480 ;--* SAUVEGARDE DE L'ORIGINE *--
7003  ED535E70  490         ld   (orgx),de
7007  226070    500         ld   (orgy),hl
                510 ;--* ORIGINE = (0,0) *--
700A  110000    520         ld   de,0
700D  210000    530         ld   hl,0
7010  CDC9BB    540         call origin
                550 ;--* TRACAGE POINT DEPART *--
7013  ED5B5A70  560         ld   de,(depx)
7017  2A5C70    570         ld   hl,(depy)
701A  CDEABB    580         call tracpi
                590 ;--* TRACAGE 1er COTE *--
701D  ED5B5670  600         ld   de,(coth)
7021  210000    610         ld   hl,0
7024  CDF9BB    620         call tracdl
                630 ;--* TRACAGE 2eme COTE *--
7027  110000    640         ld   de,0
702A  2A5870    650         ld   hl,(cotv)
702D  CDF9BB    660         call tracdl
                670 ;--* CALCUL DE - COTH *--
7030  2A5670    680         ld   hl,(coth)
7033  7C        690         ld   a,h
7034  2F        700         cpl
7035  67        710         ld   h,a
7036  7D        720         ld   a,l
7037  2F        730         cpl
7038  6F        740         ld   l,a
7039  23        750         inc  hl
                760 ;--* TRACAGE 3eme COTE *--
703A  EB        770         ex   de,hl
703B  210000    780         ld   hl,0
703E  CDF9BB    790         call tracdl
                800 ;--* TRACAGE 4eme COTE *--
7041  ED5B5A70  810         ld   de,(depx)
7045  2A5C70    820         ld   hl,(depy)
7048  CDF6BB    830         call tracd2
                840 ;--* REMISE ORIGINE INITIALE *--
704B  ED5B5E70  850         ld   de,(orgx)
704F  2A6070    860         ld   hl,(orgy)
7052  CDC9BB    870         call origin
7055  C9        880         ret
                890 ;
7056            900 coth:   defs 2                  ;COTE HORIZONTAL
7058            910 cotv:   defs 2                  ;COTE VERTICAL
705A            920 depx:   defs 2                  ;DEPART X
```

```
705C            930 depy:   defs 2                  ;DEPART Y
705E            940 orgx:   defs 2                  ;STOCK ORIGINE X
7060            950 orgy:   defs 2                  ;STOCK ORIGINE Y


Pass 2 errors: 00


coth   7056  cotv   7058  depx   705A
depy   705C  ext    6FAA  nom    6FD8
orgx   705E  orgy   7060  origin BBC9
rechor BBCC  rout   7000  rsx    BCD1
space  6FE2  tracd1 BBF9  tracd2 BBF6
tracpi BBEA


Table used:   197  from   900
Executes: 28576
```

TRACÉ DE RECTANGLE (avec orientation)

Fonction : tracé d'un rectangle quelconque.

Les routines précédentes nous ont permis de nous familiariser avec certaines procédures graphiques comme le traitement des origines, le tracé de points ou de droites.

Cependant, force est d'avouer que la routine de rectangle est très limitative, dans la mesure où seuls des rectangles dont les côtés sont parallèles aux bords de l'écran peuvent être tracés. Il serait donc très intéressant de pouvoir construire des rectangles de n'importe quelle orientation dans l'écran, repérée par l'angle a que fait le premier côté par rapport à l'horizontale.

Ainsi, supposons que le point de départ devienne l'origine de coordonnée 0,0 ; de même les deux côtés portent les noms COT1 et COT2, et l'angle d'inclinaison est a ; alors les coordonnées des trois autres sommets seraient respectivement et relativement par rapport au précédent : COT1×COS a, COT1×SIN a ; — COT2×SIN a, COT2×COS a et — COT1×COS a, — COT1×SIN a.

Cela nécessitera donc l'emploi des vecteurs mathématiques. De plus, le traitement des lignes trigonométriques ne pouvant se faire qu'en virgule flottante, il faudra convertir l'angle et les côtés ; puis effectuer les calculs et reconvertir les résultats en entier puisqu'ils serviront de coordonnées.

Syntaxe : |RECTANGLE,X,Y,A,B,a

avec X : coordonnée X du point de départ du rectangle.
X compris entre 0 et 639.
Y : coordonnée Y du point de départ du rectangle.
Y compris entre 0 et 399.
A : premier côté.
B : deuxième côté.
a : angle d'inclinaison du premier côté en degrés.

Si une fenêtre graphique est définie, seuls les points inclus dans cette fenêtre seront tracés.

Routines système utilisées

Les adresses fournies dans la liste assembleur sont celles du CPC 6128. Pour les vecteurs mathématiques sont indiquées entre parenthèses les adresses correspondant au CPC 464. Les autres adresses restent inchangées quel que soit le CPC employé.

#BDAC (#BD88) : calcul du sinus d'un angle.
#BDAF (#BD8B) : calcul du cosinus d'un angle.
#BD61 (#BD3D) : copie d'un nombre d'une adresse à une autre.
#BD64 (#BD40) : conversion ENTIER en FLOTTANT (pointé par DE).
#BD67 (#BD43) : conversion ENTIER en FLOTTANT (pointé par HL).
#BD6D (#BD49) : conversion FLOTTANT en ENTIER.
#BD85 (#BD61) : multiplication en FLOTTANT.
#BD94 (#BD70) : test du signe d'un nombre en FLOTTANT.

#B113 (#B8F7) : pointeur indicateur de l'unité d'angle.

#BBC9 : positionnement de l'origine.
#BBCC : recherche de l'origine en cours.
#BBEA : tracé d'un point.
#BBF6 : tracé d'une droite avec coordonnées absolues.
#BBF9 : tracé d'une droite avec coordonnées relatives.

Explications

Les paramètres sont tout d'abord stockés dans DEPX et DEPY pour les coordonnées du point de départ, dans COT1CO et COT2CO pour les deux côtés, et dans ANG pour l'angle.

Trois sous-programmes sont élaborés. Le premier, appelé CONV, permet de faire la conversion d'un côté en virgule flottante et de le stocker à deux adresses différentes. Le suivant, ENT, sert à effectuer la conversion inverse ; c'est-à-dire de flottant en entier, en testant auparavant le signe afin de le rétablir dans l'expression du nombre entier (en effet la routine #BD6D fournit toujours un résultat positif). Enfin, NEGAT, le troisième sous-programme, effectue la négation d'un entier selon la même méthode que dans le programme précédent, à savoir par complément à deux.

La routine débute par la recherche et le stockage de l'origine en cours dans les variables ORGX et ORGY. La nouvelle origine devient le point de départ repéré par ses coordonnées DEPX et DEPY.

Les côtés sont alors convertis en flottant dans COT1CO et COT1SI d'une part, et dans COT2CO et COT2SI d'autre part, ceci à l'aide du sous-programme CONV. De même pour l'angle, converti dans STO1 et STO2.

On fixe ensuite l'unité d'angle en degrés par l'envoi de 255 (#FF) dans le pointeur #B113.

La routine effectue la multiplication de chaque côté par le sinus et le cosinus de l'angle a, ce qui explique le besoin que nous avions de stocker chaque côté deux fois. Chacun de ces quatre calculs est alors transformé en nombre entier aux mêmes adresses : COT1CO contient le côté 1 multiplié par le cosinus de l'angle a, et ce, en entier, COT1SI le côté 1 multiplié par le sinus de l'angle a, et ainsi de suite.

On peut alors tracer les quatre côtés, après avoir placé le point de départ grâce à la routine #BBEA.

Les trois premiers côtés sont tracés par appel de la routine #BBF9, qui demande les coordonnées du point d'arrivée de la droite relativement au point de départ. Il est donc nécessaire pour les deuxième et troisième côtés de passer par le sous-programme NEGAT.

Le dernier côté est tracé par la routine #BBF6 qui relie le dernier point au point de départ du rectangle. La routine s'achève par le rétablissement de l'origine initiale.

Démonstration

Le programme BASIC de démonstration utilise la routine sous sa forme RSX. Elle permet d'effectuer une série de rotations d'un rectangle.

```
10 MODE 2
20 FOR I=0 TO 30 STEP 2
30 |RECTANGLE,100+i,200-4×I,120,80,90-3×I
40 NEXT
```

Pour I=0, a = 90 degrés, le rectangle aura son plus grand côté dans le sens vertical. Petit à petit, en suivant I, l'angle a diminué. Pour I=30, a vaut 0. Le rectangle possède donc maintenant son plus grand côté dans le sens horizontal.

Noter qu'en utilisant la routine COULGRAF ou, pour le CPC 6128, les ordres BASIC GRAPHICS PEN et GRAPHICS PAPER, on peut obtenir l'illusion d'une animation dans laquelle un objet rectangulaire chuterait en tournant.

Pour le CPC 6128, il faut rajouter :

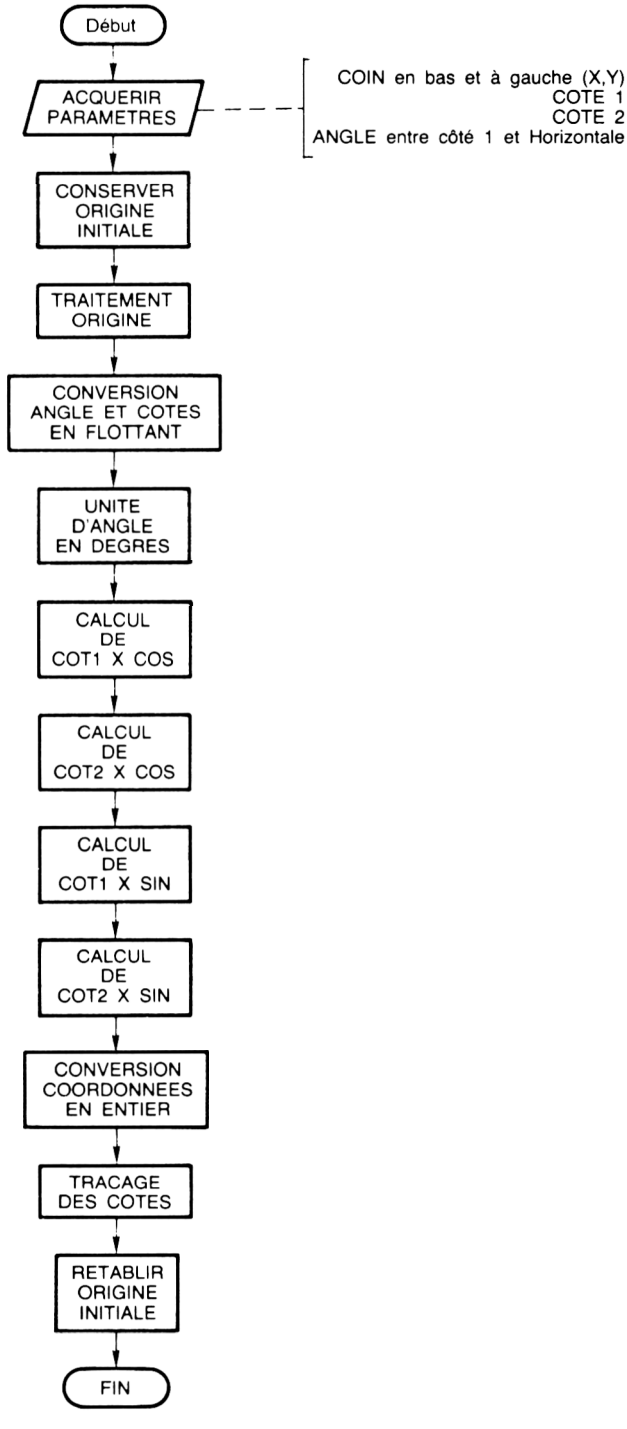
```
15 GRAPHICS PAPER 0
25 GRAPHICS PEN 1
32 GRAPHICS PEN 0
34 |RECTANGLE,100+I,200-4×I,120,80,90-3×I
```

Pour le CPC 464, les rajouts sont :

```
25 |COULGRAF,1,0
32 |COULGRAF,0,0
34 |RECTANGLE,100+I,200-4×I,120,80,90-3×I
```

Attention, si vous utilisez les deux routines |RECTANGLE et |COULGRAF en même temps, il faut veiller à ce que leurs adresses d'implantation soient bien compatibles.

Routine de tracé de rectangle avec orientation



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

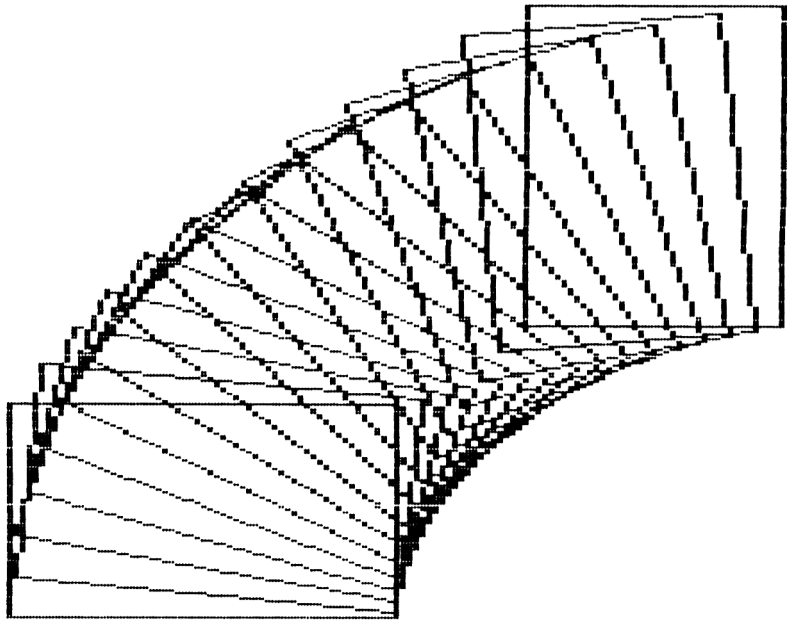
```
                 10 ;=*  ROUTINE RECTANGLE 2 CPC 6128 *--
                 20 ;
6FA0             30         ent  #6fa0
6FA0             40         org  #6fa0
                 50 ;
                 60 ;--* TRAITEMENT RSX *--
                 70 ;
BCD1             80 rsx:    equ  #bcd1
6FA0  01AA6F     90         ld   bc,ext
6FA3  21ED6F    100         ld   hl,space
6FA6  CDD1BC    110         call rsx
6FA9  C9        120         ret
6FAA  E36F      130 ext:    defw nom
                140 ;--* RECUPERATION DES PARAMETRES *--
6FAC  DD5601    150         ld   d,(ix+1)
6FAF  DD5E00    160         ld   e,(ix+0)
6FB2  ED53E770  170         ld   (ang),de
6FB6  DD4603    180         ld   b,(ix+3)
6FB9  DD4E02    190         ld   c,(ix+2)
6FBC  ED43FB70  200         ld   (cot2co),bc
6FC0  DD4605    210         ld   b,(ix+5)
6FC3  DD4E04    220         ld   c,(ix+4)
6FC6  ED43F170  230         ld   (cot1co),bc
6FCA  DD4607    240         ld   b,(ix+7)
6FCD  DD4E06    250         ld   c,(ix+6)
6FD0  ED43EF70  260         ld   (depy),bc
6FD4  DD4609    270         ld   b,(ix+9)
6FD7  DD4E08    280         ld   c,(ix+8)
6FDA  ED43ED70  290         ld   (depx),bc
6FDE  FE05      300         cp   5
6FE0  C0        310         ret  nz
6FE1  181D      320         jr   rout
                330 ;
6FE3  52454354  340 nom:    defm "RECTANGL"
6FEB  C5        350         defb "E"+#80
6FEC  00        360         defb 0
6FED            370 space:  defs 4
                380 ;
                390 ;
                400 ;--* TRAITEMENT CALL *--
                410 ;
7000            420         org  #7000
                430 ;
BBCC            440 rechor: equ  #bbcc
```

```
BBC9                450 origin: equ  #bbc9
BD64                460 entflo: equ  #bd64
BD61                470 copie:  equ  #bd61
BD67                480 entfl2: equ  #bd67
BDAF                490 cosin:  equ  #bdaf
BD85                500 multip: equ  #bd85
BDAC                510 sinus:  equ  #bdac
B113                520 degres: equ  #b113
BBEA                530 tracpi: equ  #bbea
BBF9                540 tracdr: equ  #bbf9
BBF6                550 tracd2: equ  #bbf6
BD94                560 signe:  equ  #bd94
BD6D                570 floent: equ  #bd6d
                    580 ;
7000  CDCCBB        590 rout:   call rechor
                    600 ;--* SAUVEGARDE DE L'ORIGINE *--
7003  ED53E970      610         ld   (orgx),de
7007  22EB70        620         ld   (orgy),hl
                    630 ;--* ORIGINE = POINT DEPART *--
700A  ED5BED70      640         ld   de,(depx)
700E  2AEF70        650         ld   hl,(depy)
7011  CDC9BB        660         call origin
                    670 ;--* CONVERSION ANGLE EN FLO *--
7014  2AE770        680         ld   hl,(ang)
7017  110571        690         ld   de,stol
701A  D5            700         push de
701B  CD64BD        710         call entflo
701E  E1            720         pop  hl
701F  110A71        730         ld   de,sto2
7022  010500        740         ld   bc,5
7025  EDB0          750         ldir
                    760 ;--* CHOIX UNITE DEGRES *--
7027  3EFF          770         ld   a,255
7029  3213B1        780         ld   (degres),a
                    790 ;--* CONVERSION COTES EN FLO *--
702C  21F170        800         ld   hl,cotlco
702F  11F670        810         ld   de,cotlsi
7032  CDC770        820         call conv
7035  21FB70        830         ld   hl,cot2co
7038  110071        840         ld   de,cot2si
703B  CDC770        850         call conv
                    860 ;--* CALCUL DE cotl * COSIN *--
703E  210571        870         ld   hl,stol
7041  CDAFBD        880         call cosin
7044  11F170        890         ld   de,cotlco
```

```
7047  EB          900          ex   de,hl
7048  CD85BD      910          call multip
                  920 ;--* CALCUL DE cot2 * COSIN *--
704B  21FB70      930          ld   hl,cot2co
704E  110571      940          ld   de,sto1
7051  CD85BD      950          call multip
                  960 ;--* CALCUL DE cot1 * SINUS *--
7054  210A71      970          ld   hl,sto2
7057  CDACBD      980          call sinus
705A  11F670      990          ld   de,cot1si
705D  EB         1000          ex   de,hl
705E  CD85BD     1010          call multip
                 1020 ;--* CALCUL DE cot2 * SINUS *--
7061  210071     1030          ld   hl,cot2si
7064  110A71     1040          ld   de,sto2
7067  CD85BD     1050          call multip
                 1060 ;--* CONVERSION COORD. EN ENTIER *--
706A  21F170     1070          ld   hl,cot1co
706D  CDD170     1080          call ent
7070  21F670     1090          ld   hl,cot1si
7073  CDD170     1100          call ent
7076  21FB70     1110          ld   hl,cot2co
7079  CDD170     1120          call ent
707C  210071     1130          ld   hl,cot2si
707F  CDD170     1140          call ent
                 1150 ;--* TRACAGE POINT DE DEPART *--
7082  110000     1160          ld   de,0
7085  210000     1170          ld   hl,0
                 1180 ;--* TRACAGE DES COTES *--
7088  CDEABB     1190          call tracpi
708B  ED5BF170   1200          ld   de,(cot1co)
708F  2AF670     1210          ld   hl,(cot1si)
7092  CDF9BB     1220          call tracdr
7095  2A0071     1230          ld   hl,(cot2si)
7098  CDDF70     1240          call negat
709B  ED5BFB70   1250          ld   de,(cot2co)
709F  EB         1260          ex   de,hl
70A0  CDF9BB     1270          call tracdr
70A3  2AF170     1280          ld   hl,(cot1co)
70A6  CDDF70     1290          call negat
70A9  EB         1300          ex   de,hl
70AA  2AF670     1310          ld   hl,(cot1si)
70AD  CDDF70     1320          call negat
70B0  CDF9BB     1330          call tracdr
70B3  110000     1340          ld   de,0
70B6  210000     1350          ld   hl,0
70B9  CDF6BB     1360          call tracd2
                 1370 ;--* REMISE ORIGINE INITIALE *--
```

```
70BC  ED5BE970  1380          ld   de,(orgx)
70C0  2AEB70    1390          ld   hl,(orgy)
70C3  CDC9BB    1400          call origin
70C6  C9        1410          ret
                1420 ;
                1430 ;--* cot EN FLO DANS cotsi ET cotco *--
70C7  3E00      1440 conv:    ld   a,0
70C9  CD67BD    1450          call entfl2
70CC  EB        1460          ex   de,hl
70CD  CD61BD    1470          call copie
70D0  C9        1480          ret
                1490 ;
                1500 ;--* CONVERSION EN ENTIER *--
70D1  CD94BD    1510 ent:     call signe
70D4  F5        1520          push af
70D5  CD6DBD    1530          call floent
70D8  F1        1540          pop  af
70D9  FE02      1550          cp   2
70DB  D4DF70    1560          call nc,negat
70DE  C9        1570          ret
                1580 ;
                1590 ;--* NEGATION D'UN ENTIER *--
70DF  7C        1600 negat:   ld   a,h
70E0  2F        1610          cpl
70E1  67        1620          ld   h,a
70E2  7D        1630          ld   a,l
70E3  2F        1640          cpl
70E4  6F        1650          ld   l,a
70E5  23        1660          inc  hl
70E6  C9        1670          ret
                1680 ;
70E7            1690 ang:     defs 2                    ;STOCK ANGLE
70E9            1700 orgx:    defs 2                    ;STOCK ORIGINE X
70EB            1710 orgy:    defs 2                    ;STOCK ORIGINE Y
70ED            1720 depx:    defs 2                    ;COORD. DEPART X
70EF            1730 depy:    defs 2                    ;COORD. DEPART Y
70F1            1740 cotlco:  defs 5                    ;COTE 1 * COSINUS
70F6            1750 cotlsi:  defs 5                    ;COTE 1 * SINUS
70FB            1760 cot2co:  defs 5                    ;COTE 2 * COSINUS
7100            1770 cot2si:  defs 5                    ;COTE 2 * SINUS
7105            1780 stol:    defs 5                    ;ZONE DE CALCUL
710A            1790 sto2:    defs 5                    ;ZONE DE CALCUL
```

```
Pass 2 errors: 00


ang    70E7  conv   70C7  copie  BD61
cosin  BDAF  cotlco 70F1  cotlsi 70F6
cot2co 70FB  cot2si 7100  degres B113
depx   70ED  depy   70EF  ent    70D1
entfl2 BD67  entflo BD64  ext    6FAA
floent BD6D  multip BD85  negat  70DF
nom    6FE3  orgx   70E9  orgy   70EB
origin BBC9  rechor BBCC  rout   7000
rsx    BCD1  signe  BD94  sinus  BDAC
space  6FED  stol   7105  sto2   710A
tracd2 BBF6  tracdr BBF9  tracpi BBEA


Table used:   405  from   900
Executes: 28576
```



TRACÉ DE CERCLE

Fonction : tracé d'un cercle de rayon quelconque.

Nous vous proposons maintenant une routine permettant de tracer des cercles pour lesquels seront précisées la valeur du rayon et la position du centre.

Si nous appelons le rayon RAY, les coordonnées X et Y du cercle par rapport à son centre sont : X = RAY × COS a et Y = RAY × SIN a avec a variant de 0 à 360 degrés.

Pour tracer le cercle, il suffit alors de faire varier a de 0 à 360 et de calculer X et Y pour chaque valeur de a. Cette méthode possède l'inconvénient d'être très longue à exécuter car les calculs devant s'effectuer en flottant, il est nécessaire de convertir l'angle et le rayon dans ce système ; puis il faudra faire les calculs et reconvertir les résultats en entier, et ce 360 fois.

Une manière de gagner du temps consiste à utiliser la symétrie du cercle par rapport à l'axe horizontal. De fait, il suffit alors de calculer les coordonnées du cercle pour a allant de 0 à 180 degrés. Puis, pour chaque valeur de X, on détermine Y. On trace le point de coordonnées X,Y. On calcule alors — Y par simple négation, ce qui permet de tracer le point de coordonnées X, —Y qui est le symétrique du point précédent par rapport à l'axe horizontal.

Il n'y a dans ce cas que 180 calculs à effectuer, soit la moitié de la méthode initiale.

Syntaxe : |CERCLE,X,Y,R

avec X : coordonnée X du centre du cercle.
X compris entre 0 et 639
Y : coordonné Y du centre du cercle.
Y compris entre 0 et 399.
R : valeur du rayon.

Seuls les points inscrits à l'intérieur de la fenêtre graphique, si celle-ci est définie, seront affichés à l'écran.

Routines système utilisées

Après les adresses pour le CPC 6128 suivent entre parenthèses les adresses pour le CPC 464.

#BD64 (#BD40) : conversion ENTIER en FLOTTANT.
#BD6A (#BD46) : conversion FLOTTANT en ENTIER.
#BD85 (#BD61) : multiplication en FLOTTANT.
#BD94 (#BD70) : test du signe d'un nombre en FLOTTANT.
#BDAC (#BD88) : calcul du sinus d'un angle.
#BDAF (#BD8B) : calcul du cosinus d'un angle.

#B113 (#B8F7) : pointeur indicateur de l'unité d'angle.

#BBC9 : positionnement de l'origine.
#BBCC : recherche de l'origine en cours.
#BBEA : tracé d'un point.

Explications

Après le stockage des paramètres dans CENTX et CENTY pour les coordonnées du centre du cercle et dans RAY pour la valeur du rayon, la routine commence par la sauvegarde de l'origine en cours ; puis mise de la nouvelle origine au centre de cercle.

Le rayon est alors converti en flottant à l'adresse STO. Les degrés sont choisis comme unités d'angle en mettant la valeur 255 dans le pointeur #B113. L'angle est initialisé à 0 dans ANG, qui servira aussi comme compteur pour aller de 0 à 180.

Commence alors le calcul des coordonnées du cercle par la conversion en flottant de la valeur de l'angle. Ceci se fait dans les zones CAL1 et CAL2 qui contiendront par la suite le résultat des calculs des coordonnées.

La valeur de X est calculée par la multiplication du cosinus de l'angle par le rayon contenu dans STO. Le signe du résultat est testé grâce à #BD94 afin d'être rétabli après la conversion en entier. CAL1 contient alors la coordonnée X signée en entier.

La détermination de la coordonnée Y se fait de la même manière, le résultat étant stocké dans CAL2.

La négation pour la conversion en entier, et par la suite pour le calcul de — Y, s'effectue grâce au sous-programme NEGAT qui utilise le complément à 2 du nombre se trouvant dans le registre HL.

La routine trace alors le point de coordonnée CAL1, CAL2 par l'emploi de #BBEA. Les coordonnées du symétrique de ce dernier point par rapport à l'axe horizontal sont calculées par la négation de la valeur Y contenue dans CAL2. Le point est ensuite tracé.

La valeur de l'angle contenue dans ANG est incrémentée de 1 et comparée à 181 afin de savoir si tous les points ont été tracés. Si ce n'est le cas, la procédure reprend les calculs jusqu'à ce que ANG atteigne la valeur 181.

La remise en place de l'origine initiale conclut la routine.

Démonstration

Le programme de démonstration utilise l'extension BASIC pour dessiner une corne obtenue par diminution progressive du rayon, accompagnée du déplacement du centre du cercle selon une courbe demi-elliptique.

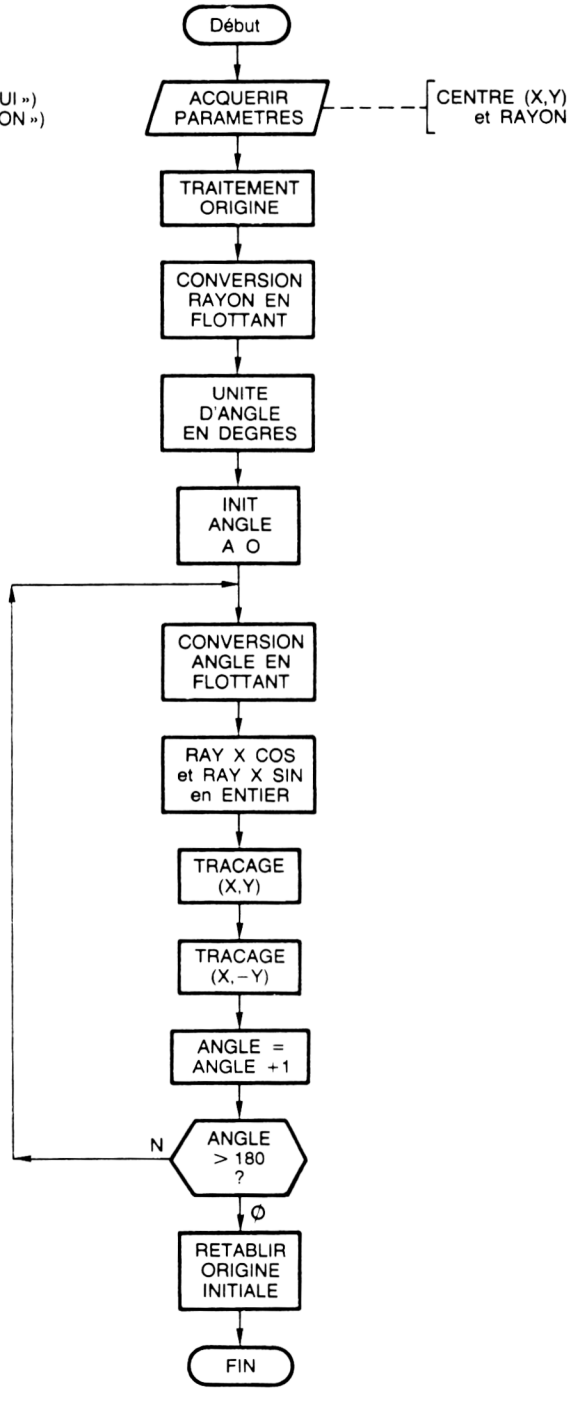
```
10 MODE 2
20 FOR I=0 TO 100 STEP 2
30|CERCLE,100+5×I,200-(I/5-10)^2,100-I
40 NEXT
```

Nous aurions pu également tracer un cercle avec un affichage beaucoup plus rapide en utilisant plus encore les propriétés de symétrie des valeurs du cercle trigonométrique. En effet, les axes X et Y apportent quelques symétries qui permettraient de réduire les calculs de SIN et COS à 45 degrés. Ceci pourrait constituer un excellent exercice.

Routine de tracé de cercle

Légende :

0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → lettre (pour « NON »)



Hisoft GENA3.1 Assembler.

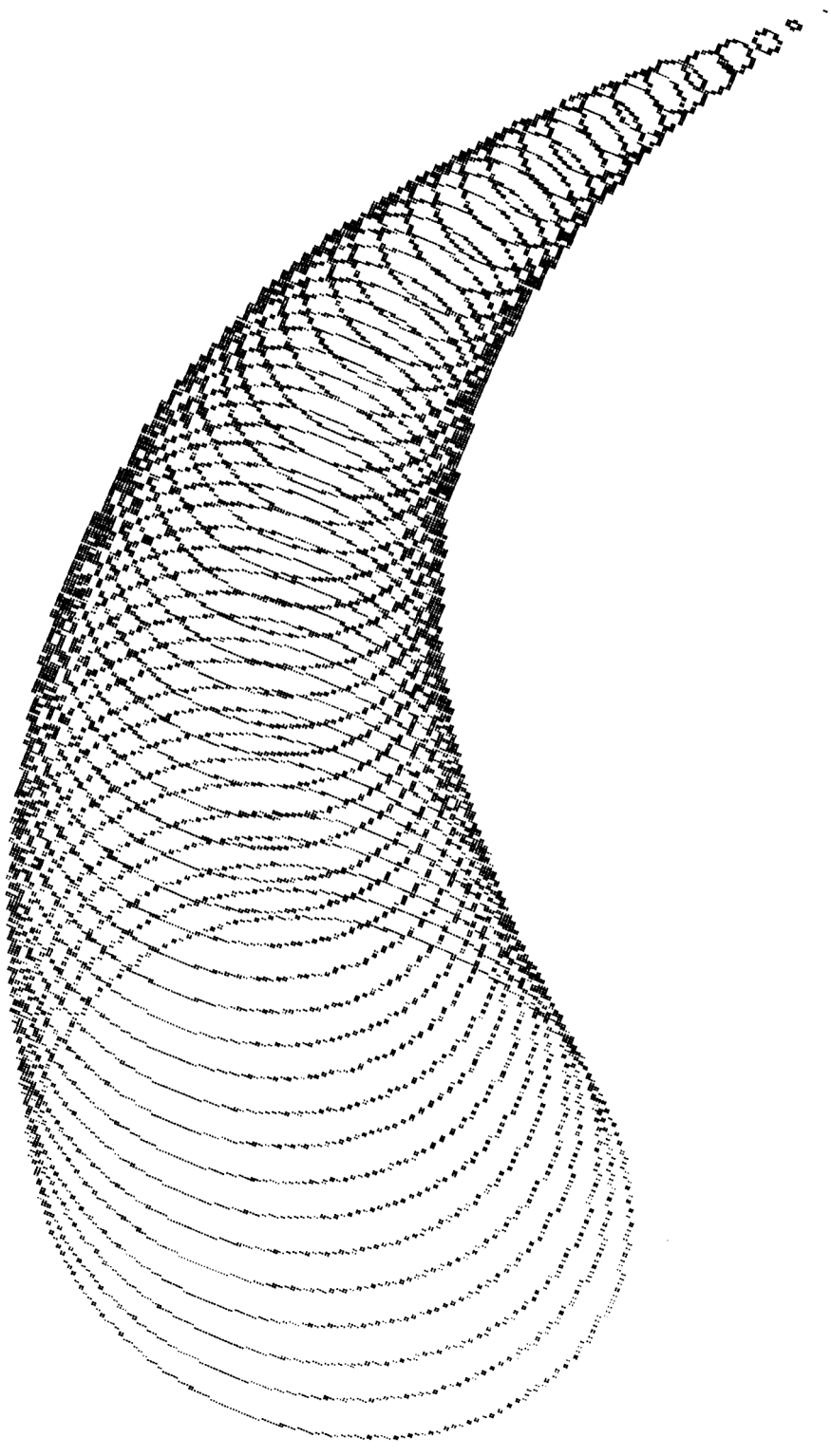
Pass 1 errors: 00

```
                  10 ;=*  ROUTINE CERCLE  CPC 6128  *=
                  20 ;
6FA0              30          ent  #6fa0
6FA0              40          org  #6fa0
                  50 ;
                  60 ;--* TRAITEMENT RSX *--
                  70 ;
BCD1              80 rsx:     equ  #bcd1
6FA0  01AA6F      90          ld   bc,ext
6FA3  21D56F     100          ld   hl,space
6FA6  CDD1BC     110          call rsx
6FA9  C9         120          ret
6FAA  CE6F       130 ext:     defw nom
                 140 ;--* RECUPERATIONS DES PARAMETRES *--
6FAC  DD4601     150          ld   b,(ix+1)
6FAF  DD4E00     160          ld   c,(ix+0)
6FB2  ED439A70   170          ld   (ray),bc
6FB6  DD5603     180          ld   d,(ix+3)
6FB9  DD5E02     190          ld   e,(ix+2)
6FBC  ED53B370   200          ld   (centy),de
6FC0  DD6605     210          ld   h,(ix+5)
6FC3  DD6E04     220          ld   l,(ix+4)
6FC6  22B170     230          ld   (centx),hl
6FC9  FE03       240          cp   3
6FCB  C0         250          ret  nz
6FCC  1832       260          jr   rout
                 270 ;
6FCE  43455243   280 nom:     defm "CERCL"
6FD3  C5         290          defb "E"+#80
6FD4  00         300          defb 0
6FD5             310 space:   defs 4
                 320 ;
                 330 ;--* TRAITEMENT CALL *--
                 340 ;
7000             350          org  #7000
                 360 ;
BBCC             370 rechor:  equ  #bbcc
BBC9             380 origin:  equ  #bbc9
BD64             390 entflo:  equ  #bd64
BDAC             400 sinus:   equ  #bdac
BDAF             410 cosin:   equ  #bdaf
BD85             420 multip:  equ  #bd85
BD94             430 signe:   equ  #bd94
BD6A             440 floent:  equ  #bd6a
BBEA             450 tracpi:  equ  #bbea
```

```
B113              460 degres: equ  #b113
                  470 ;
7000  CDCCBB      480 rout:   call rechor
                  490 ;--* SAUVEGARDE DE L'ORIGINE *--
7003  ED53AB70    500         ld   (orgx),de
7007  22AD70      510         ld   (orgy),hl
                  520 ;--* ORIGINE = CENTRE CERCLE *--
700A  ED5BB170    530         ld   de,(centx)
700E  2AB370      540         ld   hl,(centy)
7011  CDC9BB      550         call origin
                  560 ;--* CONVERSION RAYON EN FLO *--
7014  2A9A70      570         ld   hl,(ray)
7017  11A670      580         ld   de,sto
701A  CD64BD      590         call entflo
                  600 ;--* CHOIX UNITES DEGRES *--
701D  3EFF        610         ld   a,255
701F  3213B1      620         ld   (degres),a
                  630 ;--* INITIAL. DE L'ANGLE A 0 *--
7022  110000      640         ld   de,0
7025  ED53AF70    650         ld   (ang),de
                  660 ;--* CONVERSION ANGLE EN FLO *--
7029  2AAF70      670 debut:  ld   hl,(ang)
702C  119C70      680         ld   de,call
702F  CD64BD      690         call entflo
7032  11A170      700         ld   de,cal2
7035  010500      710         ld   bc,5
7038  EDB0        720         ldir
                  730 ;--* CALCUL DU COSINUS *--
703A  219C70      740         ld   hl,call
703D  CDAFBD      750         call cosin
                  760 ;--* MULTIP. RAYON * COSINUS *--
7040  11A670      770         ld   de,sto
7043  CD85BD      780         call multip
                  790 ;--* TEST SIGNE DU RESULTAT *--
7046  CD94BD      800         call signe
7049  F5          810         push af
                  820 ;--* CONVERSION EN ENTIER *--
704A  CD6ABD      830         call floent
704D  F1          840         pop  af
704E  FE02        850         cp   2
7050  D49270      860         call nc,negat
7053  229C70      870 suite:  ld   (call),hl
                  880 ;--* CALCUL DU SINUS *--
7056  21A170      890         ld   hl,cal2
7059  CDACBD      900         call sinus
                  910 ;--* MULTIP. RAYON * SINUS *--
705C  11A670      920         ld   de,sto
705F  CD85BD      930         call multip
                  940 ;--* CONVERSION EN ENTIER *--
```

```
7062  CD6ABD     950          call floent
                 960 ;--* TRACAGE DE (X,Y) *--
7065  22A170     970          ld   (cal2),hl
7068  ED5B9C70   980          ld   de,(call)
706C  D5         990          push de
706D  E5        1000          push hl
706E  CDEABB    1010          call tracpi
7071  E1        1020          pop  hl
7072  D1        1030          pop  de
                1040 ;--* CALCUL DE - Y *--
7073  CD9270    1050          call negat
                1060 ;--* TRACAGE DE (X,-Y) *--
7076  CDEABB    1070          call tracpi
                1080 ;--* INCREMENTATION ET TEST ANGLE *--
7079  2AAF70    1090          ld   hl,(ang)
707C  23        1100          inc  hl
707D  22AF70    1110          ld   (ang),hl
7080  11B500    1120          ld   de,181
7083  ED52      1130          sbc  hl,de
7085  20A2      1140          jr   nz,debut
                1150 ;--* REMISE ORIGINE INITIALE *--
7087  ED5BAB70  1160          ld   de,(orgx)
708B  2AAD70    1170          ld   hl,(orgy)
708E  CDC9BB    1180          call origin
7091  C9        1190          ret
                1200 ;
                1210 ; --* NEGATION D'UN ENTIER *--
7092  7C        1220 negat:   ld   a,h
7093  2F        1230          cpl
7094  67        1240          ld   h,a
7095  7D        1250          ld   a,l
7096  2F        1260          cpl
7097  6F        1270          ld   l,a
7098  23        1280          inc  hl
7099  C9        1290          ret
                1300 ;
709A            1310 ray:     defs 2                    ;STOCKAGE RAYON
709C            1320 call:    defs 5                    ;ZONE DE CALCUL
70A1            1330 cal2:    defs 5                    ;ZONE DE CALCUL
70A6            1340 sto:     defs 5                    ;STOCK RAY FLO
70AB            1350 orgx:    defs 2                    ;STOCK ORIGINE X
70AD            1360 orgy:    defs 2                    ;STOCK ORIGINE Y
70AF            1370 ang:     defs 2                    ;STOCKAGE ANGLE
70B1            1380 centx:   defs 2                    ;CENTRE X CERCLE
70B3            1390 centy:   defs 2                    ;CENTRE Y CERCLE
```

```
Pass 2 errors: 00


ang    70AF  call   709C  cal2   70A1
centx  70B1  centy  70B3  cosin  BDAF
debut  7029  degres B113  entflo BD64
ext    6FAA  floent BD6A  multip BD85
negat  7092  nom    6FCE  orgx   70AB
orgy   70AD  origin BBC9  ray    709A
rechor BBCC  rout   7000  rsx    BCD1
signe  BD94  sinus  BDAC  space  6FD5
sto    70A6  suite  7053  tracpi BBEA


Table used:   327  from   900
Executes: 28576
```



TRACÉ D'ELLIPSE (avec orientation)

Fonction : tracé d'une ellipse.

Après le cercle, la logique impose de passer aux ellipses. En effet, une ellipse n'est qu'un cercle déformé. Rappelons nous ; les coordonnées d'un cercle sont obtenues par les formules : X = RAYON × COS c et Y = RAYON × SIN c (c variant de 0 à 360). Pour l'ellipse, il suffit de fixer les valeurs A et B des deux demi-axes perpendiculaires. Les formules deviennent alors : X = A × COS c et Y = B × SIN c, c variant toujours de 0 à 360.

En fonction de cela, nous aurions pu reprendre la routine de cercle et modifier le calcul des coordonnées en remplaçant RAY par A ou B selon le cas, le reste étant identique. Nous avons préféré, plutôt que de réécrire une routine pratiquement semblable à la précédente, inclure la possibilité de tracer des ellipses faisant un angle b avec l'horizontale, c'est-à-dire pouvoir orienter celles-ci dans le plan.

Pour cela, il sera nécessaire de fournir à la routine, outre les paramètres courants, coordonnées du centre et valeurs des deux demi-axes, un paramètre supplémentaire fixant l'angle du premier demi-axe avec l'horizontale.

Celui-ci sera facultatif, avec mise par défaut à 0, ce qui trace des ellipses dont les axes sont parallèles aux bords de l'écran.

Le calcul des coordonnées s'effectuera d'abord, pour chaque valeur de c (de 0 à 360), sans tenir compte de l'orientation désirée de l'ellipse. En effet, l'ellipse orientée d'un angle a correspond à une ellipse aux axes parallèles aux bords de l'écran, qui aurait pivoté autour de son centre, et ce, d'un angle de rotation d'une valeur a, ce qui veut dire que tous ses points auraient pivoté du même angle.

Si X1 et Y1 sont les coordonnées d'un point dans un repère donné, X et Y les coordonnées de ce même point dans un nouveau repère ayant effectué une rotation ROT par rapport à l'ancien, alors : X = X1 COS ROT + Y1 SIN ROT et Y = X1 SIN ROT − Y1 COS ROT. Après avoir calculé chaque point de l'ellipse non orientée, nous appliquerons ces formules pour obtenir les coordonnées finales.

Syntaxe : |ELLIPSE,X,Y,A,B,[a]

avec X : coordonnée X du centre de l'ellipse.
X compris entre 0 et 639.
Y : coordonnée Y du centre de l'ellipse.
Y compris entre 0 et 399.
A : longueur du premier demi-axe.
B : longueur du deuxième demi-axe.

a : angle d'inclinaison de l'ellipse par rapport à l'horizontale, à droite du centre et vers le haut (0 par défaut).

Routines système utilisées

Entre parenthèses, vous trouverez les adresses relatives au CPC 464, les premières étant valables pour le CPC 6128.

#BD61 (#BD3D) : copie nombre en FLOTTANT.
#BD64 (#BD40) : conversion ENTIER en FLOTTANT.
#BD6A (#BD46) : conversion FLOTTANT en ENTIER.
#BD7C (#BD58) : addition en FLOTTANT.
#BD82 (#BD5B) : soustraction en FLOTTANT.
#BD85 (#BD61) : multiplication en FLOTTANT.
#BD94 (#BD70) : test du signe d'un nombre en FLOTTANT.
#BDAC (#BD88) : calcul du sinus d'un angle.
#BDAF (#BD8B) : calcul du cosinus d'un angle.

#B113 (#B8F7) : pointeur indicateur de l'unité d'angle.

#BBC9 : positionnement de l'origine.
#BBCC : recherche de l'origine en cours.
#BBEA : tracé d'un point.

Explications

La routine débute par le stockage des paramètres : dans CENTX et CENTY pour les coordonnées du centre de l'ellipse, dans AXE1 et AXE2 pour les demi-axes, et enfin dans ROT pour l'angle d'inclinaison si celui-ci est défini.

On effectue ensuite la recherche de l'origine en cours, sa sauvegarde dans ORGX et ORGY et le positionnement de la nouvelle origine au centre de l'ellipse.

L'unité d'angle choisie sera le degré par la mise de 255 dans le pointeur #B113. L'angle de rotation est alors converti en flottant dans les zones ROTCOS et ROTSIN, puis dans ces mêmes zones sont placées les valeurs de son cosinus et de son sinus. (Pour le CPC 464, rajouter XOR A avant les deux CALL ENTFLO.) Les deux axes sont à leur tour convertis en flottant dans STO1 et STO2 avant que l'angle soit initialisé à 0 dans ANG.

La routine démarre alors véritablement par la conversion de l'angle en flottant. Cette valeur est stockée dans CAL1 et CAL2 qui seront en fait des zones de calculs. On cherche ensuite les coordonnées (X1 et Y1) de l'ellipse non orientée par la multiplication de AXE1 par le cosinus de l'angle et par celle de AXE2 par le sinus. X1 sera conservé dans CAL1 et CAL3, Y1 dans CAL2 et CAL4.

Pour calculer les coordonnées X et Y de l'ellipse ayant effectuée une rotation d'un angle ROT, il suffit alors d'appliquer les formules vues précédemment. X et Y sont convertis en entier par le même type de sous-programme déjà utilisé dans la routine CERCLE qui fait appel à un autre sous-programme permettant le calcul de la négation d'un nombre.

Le point est donc tracé en X,Y par la routine #BBEA. L'angle est incrémenté de 1 et stocké à nouveau dans ANG, puis il est soustrait à 361 afin de vérifier si tous les points ont bien été tracés. Dans le cas contraire, l'exécution reprend jusqu'à ce que l'angle atteigne la valeur 361.

Noter la présence de l'instruction JR NZ,PONT pour le branchement au début de la routine de calcul des coordonnées. Il aurait fallu écrire JR NZ,DEBUT, mais comme il y a plus de 126 octets entre l'instruction et l'emplacement du branchement, cela est impossible. Plutôt que d'utiliser l'instruction JP NZ, DEBUT, qui si elle effectue le même travail ne permet pas de reloger la routine, nous avons préféré placer un relais, appelé PONT, qui ramène à DEBUT mais en deux étapes.

La routine se termine par la remise de l'origine initiale.

Démonstration

Nous vous proposons deux programmes de démonstration utilisant chacun cette routine sous forme d'extension du BASIC.

Le premier utilise le tracé d'une série d'ellipses non orientées permettant d'obtenir finalement un cône. Pour cela, il faut jouer sur la position du centre des ellipses et faire varier chacun des deux axes, le premier augmentant au fur et à mesure que le deuxième diminue.

```
10 MODE 2
20 FOR I=0 TO 45
30 |ELLIPSE,100+5×I,190-3×I,80+4×I,190-4×I
40 NEXT
```

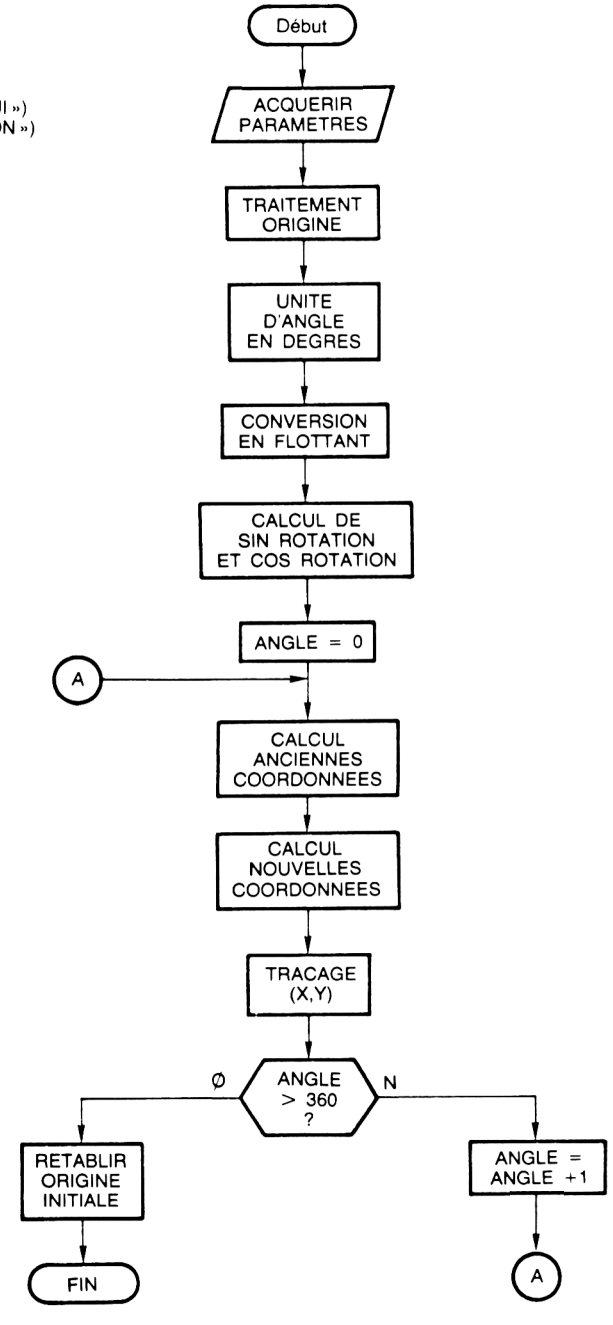
Le deuxième programme utilise l'option orientation :

```
10 MODE 2
20 FOR I=0 TO 90 STEP 10
30 |ELLIPSE,320,200,180,80,i
40 NEXT
```

Routine de tracé d'ellipse orientable

Légende :

0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → lettre (pour « NON »)



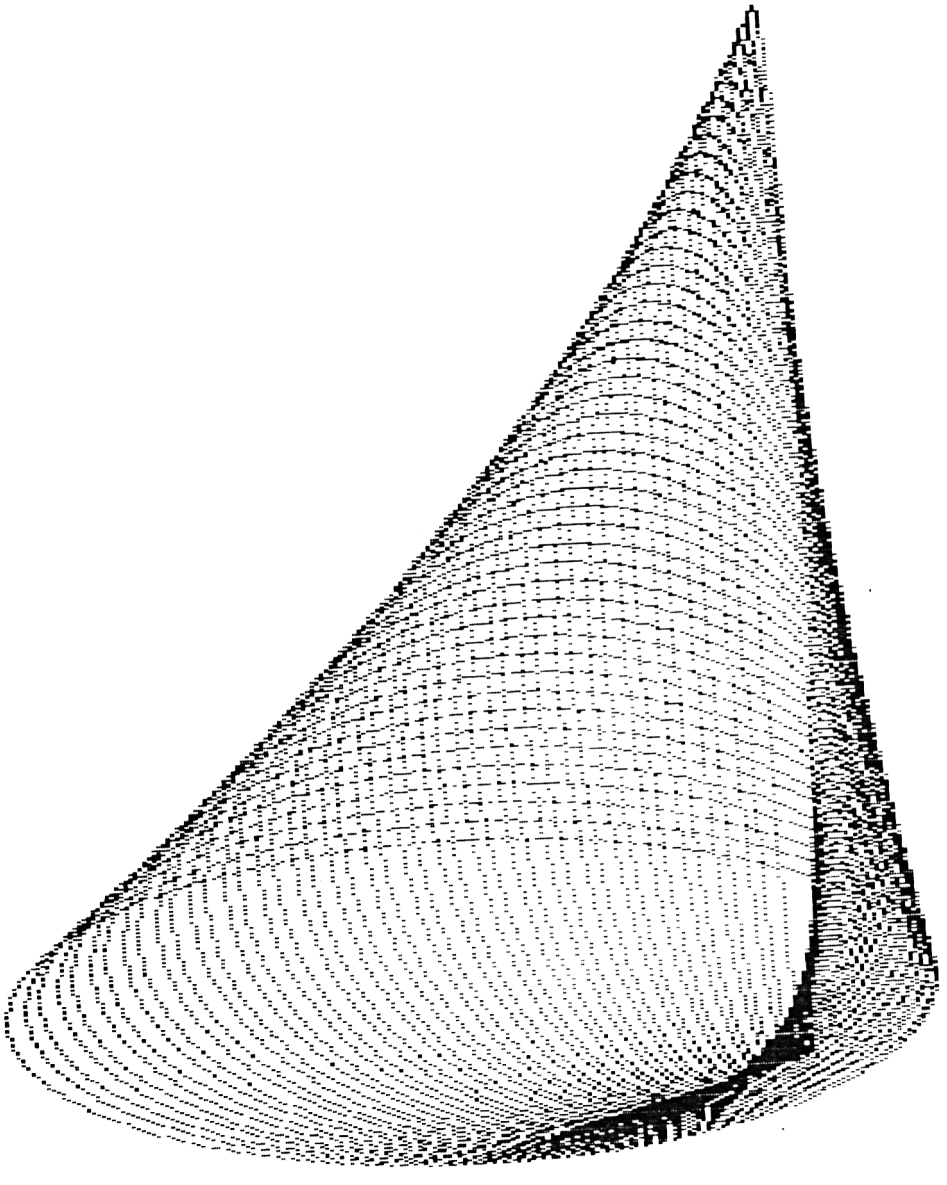
```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                 10 ;=*  ROUTINE ELLIPSE CPC 6128  *=
                 20 ;
6F90             30          ent  #6f90
6F90             40          org  #6f90
                 50 ;
                 60 ;--* TRAITEMENT RSX *--
                 70 ;
BCD1             80 rsx:     equ  #bcd1
6F90  019A6F     90          ld   bc,ext
6F93  21EA6F    100          ld   hl,space
6F96  CDD1BC    110          call rsx
6F99  C9        120          ret
6F9A  E26F      130 ext:     defw nom
                140 ;--* RECUPERATION DES PARAMETRES *--
                150 ;--* TEST SI 5 PARAMETRES *--
6F9C  FE05      160          cp   5
6F9E  2010      170          jr   nz,p1
6FA0  DD4601    180          ld   b,(ix+1)
6FA3  DD4E00    190          ld   c,(ix+0)
6FA6  ED432A71  200          ld   (rot),bc
6FAA  DD23      210          inc  ix
6FAC  DD23      220          inc  ix
6FAE  1808      230          jr   p2
                240 ;--* TEST SI 4 PARAMETRES *--
6FB0  FE04      250 p1:      cp   4
6FB2  C0        260          ret  nz
6FB3  212A71    270          ld   hl,rot
6FB6  3600      280          ld   (hl),0
6FB8  DD5605    290 p2:      ld   d,(ix+5)
6FBB  DD5E04    300          ld   e,(ix+4)
6FBE  DD4607    310          ld   b,(ix+7)
6FC1  DD4E06    320          ld   c,(ix+6)
6FC4  ED532871  330          ld   (centy),de
6FC8  ED432671  340          ld   (centx),bc
6FCC  DD5601    350          ld   d,(ix+1)
6FCF  DD5E00    360          ld   e,(ix+0)
6FD2  DD4603    370          ld   b,(ix+3)
6FD5  DD4E02    380          ld   c,(ix+2)
6FD8  ED43FE70  390          ld   (axe1),bc
6FDC  ED530071  400          ld   (axe2),de
6FE0  181E      410          jr   rout
                420 ;
6FE2  454C4C49  430 nom:     defm "ELLIPS"
6FE8  C5        440          defb "E"+#80
6FE9  00        450          defb 0
6FEA            460 space:   defs 4
```

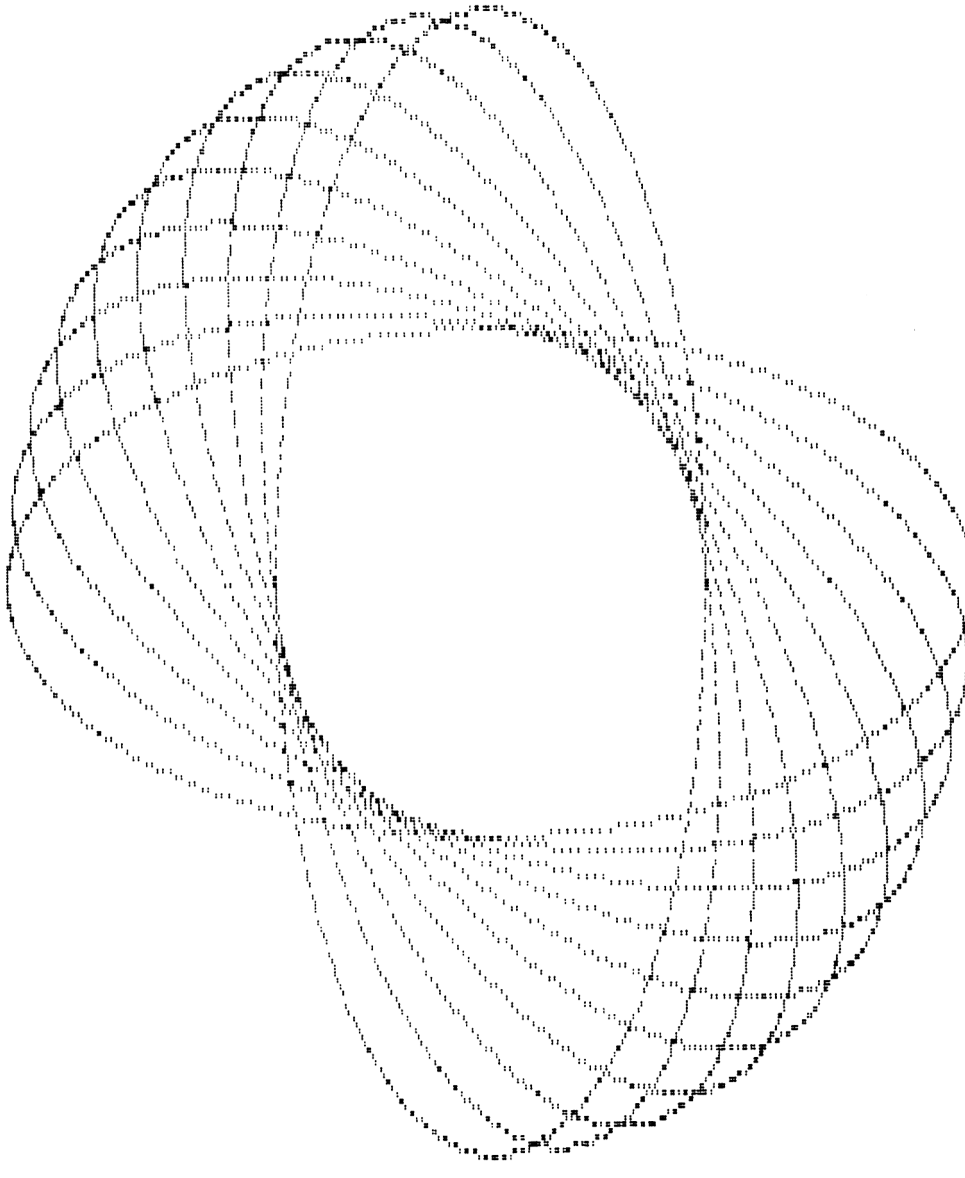
```
                470 ;
                480 ;
                490 ;--* TRAITEMENT CALL *--
                500 ;
7000            510         org  #7000
                520 ;
BBCC            530 rechor: equ  #bbcc
BBC9            540 origin: equ  #bbc9
BD61            550 copie:  equ  #bd61
BD64            560 entflo: equ  #bd64
BDAF            570 cosin:  equ  #bdaf
BDAC            580 sinus:  equ  #bdac
BD7C            590 additi: equ  #bd7c
BD82            600 soustr: equ  #bd82
BD85            610 multip: equ  #bd85
BD94            620 signe:  equ  #bd94
BD6A            630 floent: equ  #bd6a
BBEA            640 tracpi: equ  #bbea
B113            650 degres: equ  #b113
                660 ;
                670 ;--* RECHERCHE ET SAUVEGARDE ORIGINE *--
7000  CDCCBB    680 rout:   call rechor
7003  ED532071  690         ld   (orgx),de
7007  222271    700         ld   (orgy),hl
                710 ;--* ORIGINE = CENTRE ELLIPSE *--
700A  ED5B2671  720         ld   de,(centx)
700E  2A2871    730         ld   hl,(centy)
7011  CDC9BB    740         call origin
                750 ;--* CHOIX UNITE DEGRES *--
7014  3EFF      760         ld   a,255
7016  3213B1    770         ld   (degres),a
                780 ;--* CONVERSION ROT EN FLO *--
7019  2A2A71    790         ld   hl,(rot)
701C  E5        800         push hl
701D  113171    810         ld   de,rotsin
7020  AF        820         xor  a
7021  CD64BD    830         call entflo
7024  E1        840         pop  hl
7025  112C71    850         ld   de,rotcos
7028  AF        860         xor  a
7029  CD64BD    870         call entflo
                880 ;--* CALCUL DU SIN ET COS DE ROT *--
702C  213171    890         ld   hl,rotsin
702F  CDACBD    900         call sinus
7032  212C71    910         ld   hl,rotcos
7035  CDAFBD    920         call cosin
                930 ;--* CONVERSION AXE1 ET AXE2 EN FLO *--
7038  2AFE70    940         ld   hl,(axe1)
703B  111671    950         ld   de,stol
```

```
703E  CD64BD     960          call entflo
7041  2A0071     970          ld   hl,(axe2)
7044  111B71     980          ld   de,sto2
7047  CD64BD     990          call entflo
                1000 ;--* INITIAL. ANGLE A 0 *--
704A  110000    1010          ld   de,0
704D  ED532471  1020          ld   (ang),de
                1030 ;--* CONVERSION ANGLE EN FLO *--
7051  3E00      1040 debut:   ld   a,0
7053  2A2471    1050          ld   hl,(ang)
7056  110271    1060          ld   de,call
7059  CD64BD    1070          call entflo
705C  110771    1080          ld   de,cal2
705F  010500    1090          ld   bc,5
7062  EDB0      1100          ldir
                1110 ;--* CALCUL DE X1 = COS ANG * AXE1 *--
                1120 ;--*        dans call et cal3       *--
7064  210271    1130          ld   hl,call
7067  CDAFBD    1140          call cosin
706A  111671    1150          ld   de,stol
706D  CD85BD    1160          call multip
7070  110C71    1170          ld   de,cal3
7073  EB        1180          ex   de,hl
7074  CD61BD    1190          call copie
7077  1802      1200          jr   al
                1210
7079  18D6      1220 pont:    jr   debut
                1230 ;--* CALCUL DE Y1 = SIN ANG * AXE2 *--
                1240 ;--*        dans cal2 et cal4       *--
707B  210771    1250 al:      ld   hl,cal2
707E  CDACBD    1260          call sinus
7081  111B71    1270          ld   de,sto2
7084  CD85BD    1280          call multip
7087  111171    1290          ld   de,cal4
708A  EB        1300          ex   de,hl
708B  CD61BD    1310          call copie
                1320 ;--* CALCUL DE X1 * COS ROT *--
708E  210271    1330          ld   hl,call
7091  112C71    1340          ld   de,rotcos
7094  CD85BD    1350          call multip
                1360 ;--* CALCUL DE X1 * SIN ROT *--
7097  210C71    1370          ld   hl,cal3
709A  113171    1380          ld   de,rotsin
709D  CD85BD    1390          call multip
                1400 ;--* CALCUL DE Y1 * SIN ROT *--
70A0  210771    1410          ld   hl,cal2
70A3  113171    1420          ld   de,rotsin
70A6  CD85BD    1430          call multip
                1440 ;--* CALCUL DE Y1 * COS ROT *--
```

```
70A9  211171    1450          ld   hl,cal4
70AC  112C71    1460          ld   de,rotcos
70AF  CD85BD    1470          call multip
                1480 ;--* CALCUL DE X *--
70B2  210271    1490          ld   hl,call
70B5  110771    1500          ld   de,cal2
70B8  CD7CBD    1510          call additi
70BB  CDE870    1520          call conver
70BE  E5        1530          push hl
                1540 ;--* CALCUL DE Y *--
70BF  210C71    1550          ld   hl,cal3
70C2  111171    1560          ld   de,cal4
70C5  CD82BD    1570          call soustr
70C8  CDE870    1580          call conver
                1590 ;--* TRACAGE DE (X,Y) *--
70CB  D1        1600          pop  de
70CC  CDEABB    1610          call tracpi
                1620 ;--* INCREMENTATION ET TEST ANGLE *--
70CF  2A2471    1630          ld   hl,(ang)
70D2  23        1640          inc  hl
70D3  222471    1650          ld   (ang),hl
70D6  116901    1660          ld   de,361
70D9  ED52      1670          sbc  hl,de
70DB  209C      1680          jr   nz,pont
                1690 ;--* REMISE ORIGINE INITIALE *--
70DD  ED5B2071  1700          ld   de,(orgx)
70E1  2A2271    1710          ld   hl,(orgy)
70E4  CDC9BB    1720          call origin
70E7  C9        1730          ret
                1740 ;
                1750 ;--* CONVERSION EN ENTIER *--
70E8  CD94BD    1760 conver: call signe
70EB  F5        1770          push af
70EC  CD6ABD    1780          call floent
70EF  F1        1790          pop  af
70F0  FE02      1800          cp   2
70F2  D4F670    1810          call nc,negat
70F5  C9        1820          ret
                1830 ;
                1840 ;--* NEGATION D'UN ENTIER *--
70F6  7C        1850 negat:   ld   a,h
70F7  2F        1860          cpl
70F8  67        1870          ld   h,a
70F9  7D        1880          ld   a,l
70FA  2F        1890          cpl
70FB  6F        1900          ld   l,a
70FC  23        1910          inc  hl
70FD  C9        1920          ret
                1930 ;
```

```
70FE          1940 axel:   defs 2          ;LONG. AXE 1
7100          1950 axe2:   defs 2          ;LONG. AXE 2
7102          1960 cal1:   defs 5          ;ZONE CALCUL
7107          1970 cal2:   defs 5          ;ZONE CALCUL
710C          1980 cal3:   defs 5          ;ZONE CALCUL
7111          1990 cal4:   defs 5          ;ZONE CALCUL
7116          2000 stol:   defs 5          ;AXE1 EN FLO
711B          2010 sto2:   defs 5          ;AXE2 EN FLO
7120          2020 orgx:   defs 2          ;ORIG. INIT. X
7122          2030 orgy:   defs 2          ;ORIG. INIT. Y
7124          2040 ang:    defs 2          ;ZONE STOCK ANG
7126          2050 centx:  defs 2          ;CENTRE X ELLIPS
7128          2060 centy:  defs 2          ;CENTRE Y ELLIPS
712A          2070 rot:    defs 2          ;ANG DE ROTATION
712C          2080 rotcos: defs 5          ;STOC DE COS ROT
7131          2090 rotsin: defs 5          ;STOC DE SIN ROT

Pass 2 errors: 00

al     707B  additi BD7C  ang    7124
axel   70FE  axe2   7100  cal1   7102
cal2   7107  cal3   710C  cal4   7111
centx  7126  centy  7128  conver 70E8
copie  BD61  cosin  BDAF  debut  7051
degres B113  entflo BD64  ext    6F9A
floent BD6A  multip BD85  negat  70F6
nom    6FE2  orgx   7120  orgy   7122
origin BBC9  p1     6FB0  p2     6FB8
pont   7079  rechor BBCC  rot    712A
rotcos 712C  rotsin 7131  rout   7000
rsx    BCD1  signe  BD94  sinus  BDAC
soustr BD82  space  6FEA  stol   7116
sto2   711B  tracpi BBEA

Table used:   486  from   900
Executes: 28560
```





TRACÉ DE POLYGONE RÉGULIER (avec orientation)

Fonction : tracé d'un polygone régulier à n côtés.

Dernier volet de cette série de figures géométriques, le polygone régulier. Mais avant, un peu de théorie. Un polygone régulier, c'est-à-dire possédant des côtés égaux, a la particularité de pouvoir être contenu dans un cercle, appelé cercle circonscrit, passant par tous les sommets de ce polygone.

De plus, le nombre de sommets correspond au nombre de côtés ; et enfin, dernière particularité, l'angle au centre du polygone entre deux sommets consécutifs est obtenu par la division de 360 par le nombre de côtés. Nous l'appelerons, le PAS.

Il devient donc facile de comprendre qu'il suffira de reprendre la routine de cercle, moyennant certaines adaptations, afin de pouvoir tracer ces polygones ; en effet les coordonnées des sommets sont des points appartenant au cercle.

On commencera alors par placer le premier point du polygone qui pourrait être, par exemple, celui obtenu pour un angle de 0. On incrémente ensuite l'angle, de la valeur PAS, et on calcule les coordonnées du cercle correspondant à ce nouvel angle. Et ainsi de suite, jusqu'à ce que l'angle fasse 360 degrés. On aura recherché un nombre de points égal au nombre de côtés. Il ne restera plus qu'à faire se rejoindre tous ces points par des droites pour avoir le tracé du polygone.

Seulement il est possible de formuler une petite remarque. En effet, avec cette méthode, le premier point obtenu est toujours placé horizontalement et à droite du centre du polygone ; ainsi un pentagone (5 côtés) aura toujours un côté vertical à gauche du centre. On ne peut le faire pivoter.

Il nous a donc paru important d'offrir la possibilité de placer le premier point du polygone avec un angle quelconque par rapport à l'horizontale. Cet angle sera fourni par le passage à la routine d'un paramètre facultatif suivant ceux, obligatoires, concernant le centre du cercle circonscrit et valeur du rayon de ce cercle.

Dans ce cas, le premier sommet du polygone correspondra au point du cercle circonscrit, dont les coordonnées sont calculées pour l'angle de départ indiqué, s'il y a lieu, dans les paramètres.

Le calcul des autres sommets reprend la méthode décrite plus haut en incrémentant l'angle de la valeur PAS puis en reliant tous les points obtenus par des droites.

Syntaxe : |POLYGONE,N,X,Y,R,[A]

avec N : nombre de côtés du polygone.
X : coordonnée X du centre du cercle circonscrit.
X compris entre 0 et 639.
Y : coordonnée Y du centre du cercle circonscrit.
Y compris entre 0 et 399.
R : rayon du cercle circonscrit.
A : angle au centre entre le premier sommet et l'horizontale.

Routines système utilisées

Après les adresses pour le CPC 6128 suivent entre parenthèses les adresses pour le CPC 464.

#BD61 (#BD3D) : copie d'un nombre en FLOTTANT.
#BD64 (#BD40) : conversion ENTIER en FLOTTANT.
#BD6A (#BD46) : conversion FLOTTANT en ENTIER.
#BD7C (#BD58) : addition en FLOTTANT.
#BD85 (#BD61) : multiplication en FLOTTANT.
#BD88 (#BD64) : division en FLOTTANT.
#BD94 (#BD70) : test du signe d'un nombre en FLOTTANT.
#BDAC (#BD88) : calcul du sinus d'un angle.
#BDAF (#BD8B) : calcul du cosinus d'un angle.

#B113 (#B8F7) : pointeur indicateur de l'unité d'angle.

#BBC9 : positionnement de l'origine.
#BBCC : recherche de l'origine en cours.
#BBEA : tracé d'un point.
#BBF6 : tracé d'une droite.

Explications

Stockage du centre dans CENTX et CENTY pour les coordonnées du centre de l'ellipse ; stockage du rayon dans RAY pour la valeur du rayon du cercle circonscrit ; stockage du nombre de côtés dans COTENT et COMPTE ; puis stockage dans ANG de l'angle du point de départ, s'il est défini ; la routine commence par le traitement de l'origine.

Le choix de l'unité d'angle est fait en mettant 255 dans le pointeur #B113. Le rayon est converti en flottant dans STO, le nombre de côtés dans COTENT et dans COMPTE, et le nombre 360 dans PAS afin de calculer celui-ci par division de 360 par le nombre de côtés.

L'angle est ensuite initialisé en flottant. Dans le cas de la recherche des coordonnées du premier sommet, on passe directement au stockage de la valeur de l'angle dans STO1 et STO2 sans avoir ajouté le pas.

STO1 est utilisé pour y placer le résultat de la multiplication du cosinus de l'angle par le rayon, alors que STO2 contiendra celui du sinus par le rayon.

Ces deux multiplications sont faites par un sous-programme chargé également de la transformation des résultats en entier ; le cas des valeurs négatives sera traité selon la méthode du complément à deux, vue par exemple, dans la routine du cercle. STO1 et STO2 contiennent alors respectivement, l'abscisse et l'ordonnée des coordonnées du sommet.

On vérifie qu'il s'agit du premier sommet en comparant le nombre de côtés, stocké dans COTENT, avec COMPTE. S'ils sont égaux, le premier sommet est tracé.

Le compteur COMPTE est alors décrémenté de 1 et on le compare avec #FF pour savoir si le dernier côté a été atteint. Si ce n'est le cas, le branchement se fait au début des calculs où la valeur du pas est ajoutée à celle de l'angle en cours.

Les coordonnées du sommet suivant sont calculées et, au lieu de tracer ce sommet, on relie celui-ci à la dernière position par une droite, en faisant appel à la routine #BBF6.

Et on recommence ainsi jusqu'à ce que le dernier côté soit tracé. La décrémentation du compteur le fait alors passer de 0 à #FF, ce qui permet à ce moment là, par comparaison avec cette valeur, de se brancher sur la fin de la routine ; puis, on effectue la remise en place de l'origine initiale.

Démonstration

Le petit programme BASIC suivant, utilisant la routine comme extension, permet d'obtenir un effet intéressant :

```
10 MODE 2
20 FOR I=0 TO 190 STEP 6
30 |POLYGONE,6,320,200,I,I
40 NEXT
```

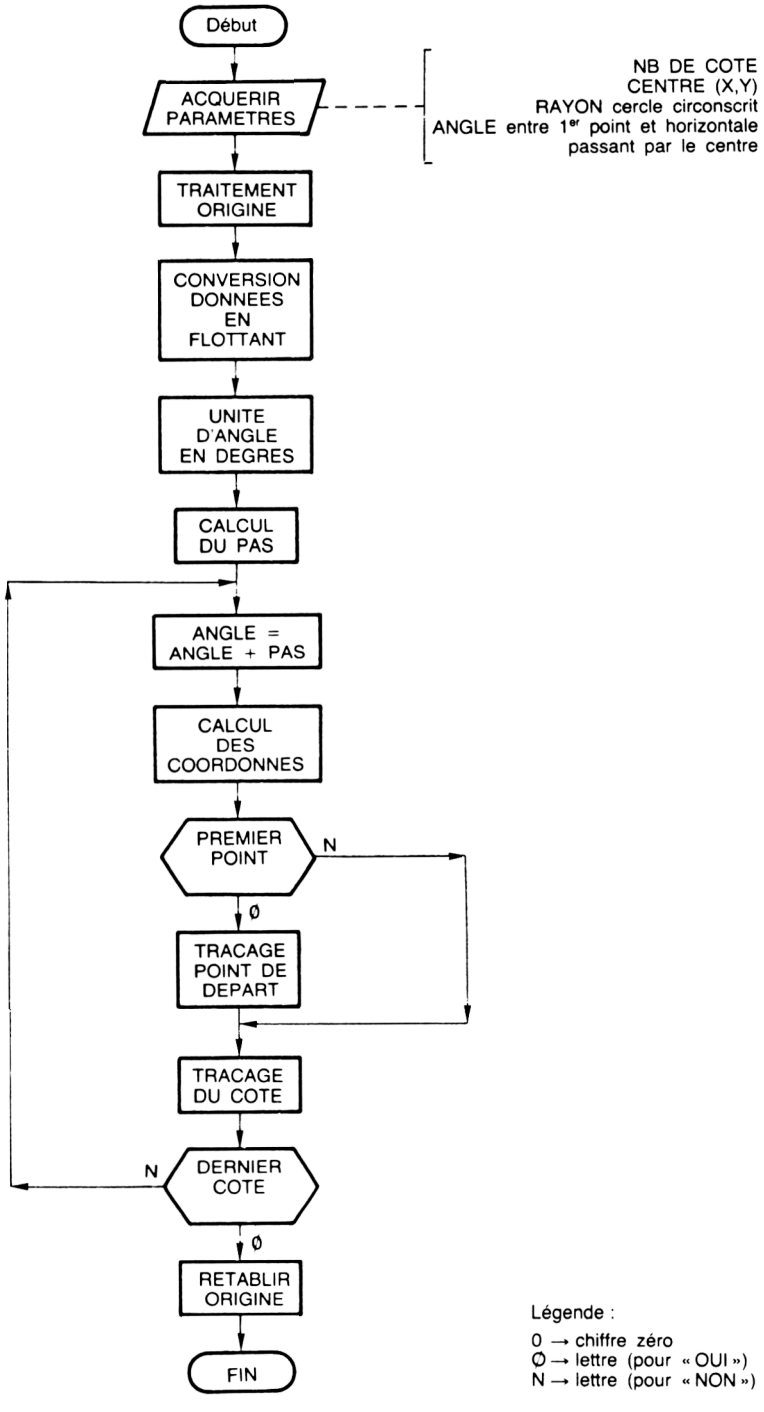
Cette fonction polygone possède également la particularité de pouvoir être utilisée pour le tracé des cercles.

En effet, si le paramètre N indiquant le nombre de côtés est important, la figure tracée est pratiquement un cercle. Pour cela, il faut, pour un rayon assez grand, donner à N une valeur de l'ordre de 30.

L'intérêt par rapport à la fonction CERCLE est d'obtenir un tracé beaucoup plus rapide, dans la mesure où seuls 30 calculs seront effectués au lieu de 180 ; comme c'était le cas dans la routine vue précédemment.

Par contre la qualité du tracé sera moindre. On pourra alors, selon l'objectif visé, utiliser pour les cercles l'une ou l'autre de ces deux routines.

Routine de tracé de polygone



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

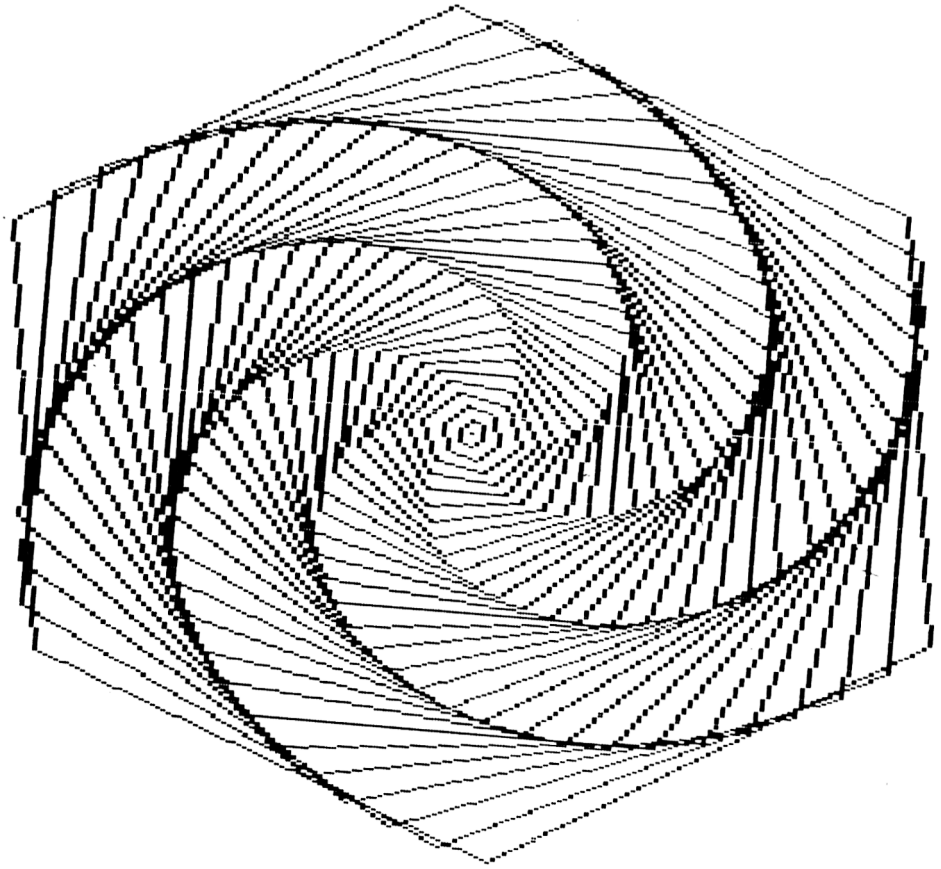
```
                 10 ;=*  ROUTINE  POLYGONE CPC 6128  *=
                 20 ;
6F90             30          ent  #6f90
6F90             40          org  #6f90
                 50 ;
                 60 ;--* TRAITEMENT RSX *--
                 70 ;
BCD1             80 rsx:     equ  #bcd1
6F90  019A6F     90          ld   bc,ext
6F93  21EE6F    100          ld   hl,space
6F96  CDD1BC    110          call rsx
6F99  C9        120          ret
6F9A  E56F      130 ext:     defw nom
                140 ;--* RECUPERATION DES PARAMETRES *--
                150 ;--* TEST SI 5 PARAMETRES *--
6F9C  FE05      160          cp   5
6F9E  2010      170          jr   nz,p1
6FA0  DD4601    180          ld   b,(ix+1)
6FA3  DD4E00    190          ld   c,(ix+0)
6FA6  ED43DD70  200          ld   (ang),bc
6FAA  DD23      210          inc  ix
6FAC  DD23      220          inc  ix
6FAE  1808      230          jr   p2
                240 ;--* TEST SI 4 PARAMETRES *--
6FB0  FE04      250 p1:      cp   4
6FB2  C0        260          ret  nz
6FB3  21DD70    270          ld   hl,ang
6FB6  3600      280          ld   (hl),0
6FB8  DD4601    290 p2:      ld   b,(ix+1)
6FBB  DD4E00    300          ld   c,(ix+0)
6FBE  ED43F270  310          ld   (ray),bc
6FC2  DD5603    320          ld   d,(ix+3)
6FC5  DD5E02    330          ld   e,(ix+2)
6FC8  ED53E470  340          ld   (centy),de
6FCC  DD6605    350          ld   h,(ix+5)
6FCF  DD6E04    360          ld   l,(ix+4)
6FD2  22E270    370          ld   (centx),hl
6FD5  DD4607    380          ld   b,(ix+7)
6FD8  DD4E06    390          ld   c,(ix+6)
6FDB  ED43F070  400          ld   (cotent),bc
6FDF  ED43F470  410          ld   (compte),bc
6FE3  181B      420          jr   rout
                430 ;
6FE5  504F4C59  440 nom:     defm "POLYGON"
6FEC  C5        450          defb "E"+#80
```

```
6FED  00          460          defb 0
6FEE              470 space:   defs 4
                  480 ;
                  490 ;
                  500 ;--* TRAITEMENT CALL *--
                  510 ;
7000              520          org  #7000
                  530 ;
BBCC              540 rechor: equ  #bbcc
BBC9              550 origin: equ  #bbc9
BBEA              560 tracpi: equ  #bbea
BD64              570 entflo: equ  #bd64
BD88              580 divise: equ  #bd88
BD7C              590 additi: equ  #bd7c
BD61              600 transf: equ  #bd61
BDAF              610 cosin:  equ  #bdaf
BDAC              620 sinus:  equ  #bdac
BBF6              630 tracot: equ  #bbf6
BD85              640 multip: equ  #bd85
BD94              650 signe:  equ  #bd94
BD6A              660 floent: equ  #bd6a
B113              670 degres: equ  #b113
                  680 ;
7000  CDCCBB      690 rout:    call rechor
                  700 ;--* SAUVEGARDE DE L'ORIGINE *--
7003  ED53D970    710          ld   (orgx),de
7007  22DB70      720          ld   (orgy),hl
                  730 ;--* ORIG. = CENTRE CERCL. CIRCONS. *--
700A  ED5BE270    740          ld   de,(centx)
700E  2AE470      750          ld   hl,(centy)
7011  CDC9BB      760          call origin
                  770 ;--* CONVERSION RAYON EN FLO *--
7014  3E00        780          ld   a,0
7016  2AF270      790          ld   hl,(ray)
7019  11D470      800          ld   de,sto
701C  CD64BD      810          call entflo
                  820 ;--* CHOIX UNITE DEGRES *--
701F  3EFF        830          ld   a,255
7021  3213B1      840          ld   (degres),a
                  850 ;--* CONVERSION COTE EN FLO *--
7024  2AF070      860          ld   hl,(cotent)
7027  11E670      870          ld   de,cote
702A  CD64BD      880          call entflo
                  890 ;--* CONVERSION 360 EN FLO *--
702D  216801      900          ld   hl,360
7030  11EB70      910          ld   de,pas
7033  CD64BD      920          call entflo
                  930 ;--* CALCUL PAS (360/cote) *--
7036  21EB70      940          ld   hl,pas
```

```
7039  11E670     950          ld   de,cote
703C  CD88BD     960          call divise
                 970 ;--* INITIAL. DE L'ANGLE *--
703F  2ADD70     980          ld   hl,(ang)
7042  11DD70     990          ld   de,ang
7045  CD64BD    1000          call entflo
7048  1809      1010          jr   al
                1020 ;--* ANG = ANG + PAS *--
704A  21DD70    1030 debut:   ld   hl,ang
704D  11EB70    1040          ld   de,pas
7050  CD7CBD    1050          call additi
                1060 ;--* ANG DANS STO1 ET STO2 *--
7053  11DD70    1070 al:      ld   de,ang
7056  21CA70    1080          ld   hl,sto1
7059  CD61BD    1090          call transf
705C  11CF70    1100          ld   de,sto2
705F  010500    1110          ld   bc,5
7062  EDB0      1120          ldir
                1130 ;--* CALCUL COSINUS ET SINUS *--
7064  21CA70    1140          ld   hl,sto1
7067  CDAFBD    1150          call cosin
706A  CDB170    1160          call multi
706D  22CA70    1170          ld   (sto1),hl
7070  21CF70    1180          ld   hl,sto2
7073  CDACBD    1190          call sinus
7076  CDB170    1200          call multi
7079  22CF70    1210          ld   (sto2),hl
                1220 ;--* TRACAGE POINT DE DEPART *--
707C  3AF070    1230          ld   a,(cotent)
707F  21F470    1240          ld   hl,compte
7082  BE        1250          cp   (hl)
7083  200C      1260          jr   nz,a2
7085  2ACF70    1270          ld   hl,(sto2)
7088  ED5BCA70  1280          ld   de,(sto1)
708C  CDEABB    1290          call tracpi
708F  180A      1300          jr   a3
                1310 ;--* TRACAGE DU COTE *--
7091  2ACF70    1320 a2:      ld   hl,(sto2)
7094  ED5BCA70  1330          ld   de,(sto1)
7098  CDF6BB    1340          call tracot
                1350 ;--* DECREMENTATION DU COMPTEUR *--
709B  3AF470    1360 a3:      ld   a,(compte)
709E  3D        1370          dec  a
709F  32F470    1380          ld   (compte),a
                1390 ;--* TEST SI DERNIER COTE *--
70A2  FEFF      1400          cp   #ff
70A4  20A4      1410          jr   nz,debut
                1420 ;--* REMISE ORIGINE INITIALE *--
70A6  ED5BD970  1430          ld   de,(orgx)
```

```
70AA  2ADB70   1440          ld   hl,(orgy)
70AD  CDC9BB   1450          call origin
70B0  C9       1460          ret
               1470 ;
               1480 ;--* MULTI. RAY * COS OU SIN *--
70B1  11D470   1490 multi:   ld   de,sto
70B4  CD85BD   1500          call multip
70B7  CD94BD   1510          call signe
70BA  F5       1520          push af
70BB  CD6ABD   1530          call floent
70BE  F1       1540          pop  af
70BF  FE02     1550          cp   2
70C1  D8       1560          ret  c
70C2  7C       1570          ld   a,h
70C3  2F       1580          cpl
70C4  67       1590          ld   h,a
70C5  7D       1600          ld   a,l
70C6  2F       1610          cpl
70C7  6F       1620          ld   l,a
70C8  23       1630          inc  hl
70C9  C9       1640          ret
               1650 ;
70CA           1660 stol:    defs 5                    ;STOCK ANG FLO
70CF           1670 sto2:    defs 5                    ;STOCK ANG FLO
70D4           1680 sto:     defs 5                    ;STOCK RAY FLO
70D9           1690 orgx:    defs 2                    ;STOCK ORIGINE X
70DB           1700 orgy:    defs 2                    ;STOCK ORIGINE Y
70DD           1710 ang:     defs 5                    ;STOCK ANGLE
70E2           1720 centx:   defs 2                    ;COORD. CENTRE X
70E4           1730 centy:   defs 2                    ;COORD. CENTRE Y
70E6           1740 cote:    defs 5                    ;NBRE COT EN FLO
70EB           1750 pas:     defs 5                    ;STOCK. DU PAS
70F0           1760 cotent:  defs 2                    ;NBRE COT EN ENT
70F2           1770 ray:     defs 2                    ;STOCK RAY ENT
70F4           1780 compte:  defs 2                    ;COMPTEUR COTES
```

```
Pass 2 errors: 00

al      7053  a2      7091  a3      709B
additi  BD7C  ang     70DD  centx   70E2
centy   70E4  compte  70F4  cosin   BDAF
cote    70E6  cotent  70F0  debut   704A
degres  B113  divise  BD88  entflo  BD64
ext     6F9A  floent  BD6A  multi   70B1
multip  BD85  nom     6FE5  orgx    70D9
orgy    70DB  origin  BBC9  pl      6FB0
p2      6FB8  pas     70EB  ray     70F2
rechor  BBCC  rout    7000  rsx     BCD1
signe   BD94  sinus   BDAC  space   6FEE
sto     70D4  sto1    70CA  sto2    70CF
tracot  BBF6  tracpi  BBEA  transf  BD61


Table used:   459  from   900
Executes: 28560
```



TRAITEMENT DE LA MÉMOIRE ÉCRAN

GÉNÉRALITÉS

Avant d'aborder toute routine concernant la mémoire écran des CPC, il est nécessaire de parfaitement connaître le mécanisme de gestion de l'écran.

Pour commencer, nous savons qu'en mode 2, il est possible d'afficher à l'écran 80 caractères par ligne, et ce sur 25 lignes. Si nous examinons un caractère de près, nous voyons qu'il s'agit en fait d'une matrice de 8 points sur 8 points. Ce qui signifie que l'écran affiche 8 × 80 = 640 points dans la longueur et dans la hauteur 8 × 25 = 200 points.

Donc, théoriquement, l'écran possède 640 × 200 points. Mais cela voudrait dire, si tous les points avaient même dimension, que la longueur serait 3 fois plus grande que la hauteur ; or ce n'est pas le cas, puisque pour rétablir les proportions, chaque point de la hauteur est dédoublé, ce qui en fait 400.

Finalement l'écran est formé de 640 × 400 points, en sachant qu'en réalité on considère 200 lignes de 2 points de haut. Chacun de ces points est repéré par ses coordonnées absolues par rapport au coin en bas à gauche de l'écran. Par exemple, le centre de l'écran a pour coordonnées : 320,200.

Pour coder un point il faut un bit, qui sera mis à 1 si le point est activé, sinon à 0. On en déduit que un octet (8 bits) permet de coder 8 points. Pour 640 points, il faudra alors 80 octets (640 = 8 × 80). Pour chaque ligne de l'écran, 80 octets seront nécessaires au codage. Comme il y a 200 lignes, il faudra 200 × 80 octets soit 16000 octets pour coder tout l'écran. Un K représentant 1024 octets, 16 K de mémoire seront réservés à l'écran dans la mémoire vive de la machine. Cette zone s'étend de l'adresse #C000 à l'adresse #FFFF.

Mais si vous avez bien suivi, vous allez me dire que 16 K font en réalité 16 × 1024 octets, c'est-à-dire 16384 octets alors qu'il n'en faut que 16000. C'est parfaitement exact ! 384 octets de cette zone sont inutilisés. Nous préciserons leur emplacement par la suite.

Comment s'organisent ces 16 K ? On pourrait penser que les octets de la mémoire se suivent pour représenter une ligne d'écran après l'autre. Ainsi la première ligne commencerait en #C000 pour se terminer 80 octets plus loin, soit en #C04F ; la deuxième ligne commencerait en #C050 pour se terminer en #C09F, et ainsi de suite jusqu'à la ligne 200.

Et bien non ! Si ce que nous venons d'envisager est valable pour la première ligne, ça ne l'est pas pour le reste. En effet, la deuxième ligne de points de l'écran commence non pas en #C050 mais en #C800. Pourquoi ? La raison en est simple. Le CPC code d'abord les premières lignes de chaque ligne de caractères puis les deuxièmes lignes de chaque ligne de caractères et ainsi jusqu'à la huitième ligne de chaque ligne de caractères.

Les premières lignes de points de chaque ligne de caractères seront alors codées de #C000 à #C7FF soit 80 × 25 octets. Cela forme donc un bloc de 2000 octets,

autrement dit de 2K. Mais 2K représentent 2048 octets et il n'en faut que 2000. Cela signifie que les 48 derniers octets de chaque bloc sont inutilisés. C'est ce que nous évoquions un peu plus haut.

De même, les deuxièmes lignes de chaque ligne de caractères seront codées de #C800 à #CFCF, laissant à nouveau les 48 derniers octets de ce bloc de 2K inutilisés. Ainsi les dernières lignes de points de chaque ligne de caractères se verront stockées de #F800 à #FFCF, les octets de #FFD0 à #FFFF étant délaissés.

En conclusion, la zone mémoire écran des CPC est formée de huit blocs de 2K, représentant chacun l'ensemble d'une des huit lignes de chaque ligne de caractères ; n'oublions pas que les 48 derniers octets sont inutilisés (48 octets × 8 blocs = 384 octets libres, chiffre cité précédemment).

Cette structure explique un phénomène que vous avez certainement déjà remarqué. Lors du chargement d'une page écran, on assiste non pas à un remplissage de l'écran ligne à ligne de haut en bas, mais à un remplissage ligne à ligne de 25 zones, réparties également sur l'écran. Chacune de ces zones correspond à une ligne de caractères.

Nous savons maintenant comment s'organise la mémoire écran des CPC. Il nous reste à expliquer le fonctionnement de cette zone lors de l'utilisation de chacun des trois modes.

Avant cela, il est nécessaire de définir un terme que nous emploierons très fréquemment par la suite. Il s'agit du terme PIXEL. Le pixel est l'unité de base de l'écran, formé selon les modes d'un nombre différent de points.

Autrement dit, c'est la plus petite partie de l'écran à laquelle il est possible de donner une couleur propre. Cette unité verra sa taille réelle augmenter au fur et à mesure que la palette des couleurs s'élargira, la résolution de l'écran étant alors moins fine.

Ainsi en mode 2, il est possible de placer 80 caractères de huit points par ligne soit 640 points. Disposant de 80 octets par ligne, chaque caractère sera codé par un octet et un bit de celui-ci correspondra à un point. Si le bit est mis à 1, le point est placé à l'écran dans la couleur de l'écriture ; s'il est à 0, le point prend la couleur du fond. L'utilisateur ne dispose alors que de deux couleurs dans ce mode. Chaque pixel représente un rectangle d'un point de large sur deux points de haut. C'est la résolution la plus fine.

En mode 1, chaque ligne permet d'afficher 40 caractères. Deux octets servent donc au codage d'un caractère, un octet codant la moitié de celui-ci. La matrice d'un caractère étant indépendante du mode, à savoir une grille de 8 × 8, 2 bits de chaque octet servent à placer un des 4 pixels de cette moitié de matrice. Or en binaire, 2 bits permettent de coder les nombres de 1 à 4 ; ce qui explique que chaque pixel, selon la valeur donnée par les 2 bits, sera affiché à l'écran dans l'une des 4 couleurs disponibles. En conclusion, mode 1 : 40 caractères de 8 pixels par ligne avec pour chaque pixel (2 points), un choix de 4 couleurs.

20 caractères peuvent être affichés, par ligne, en mode 0. Il faut alors 4 octets pour chacun d'eux. Un octet est donc utilisé pour afficher 2 pixels sur les 8 de la matrice. Un pixel nécessite dans ce cas 4 bits, mais grâce à ceux-ci peuvent être codés, en binaire, les nombres de 1 à 16. Vous comprenez alors pourquoi, par pixel, l'utilisateur dispose d'une palette de 16 couleurs. Finalement, mode 0 : 20 caractères de 8 pixels par ligne avec par pixel (4 points), un choix parmi 16 couleurs.

Le tableau suivant résume ce que nous venons de voir :

	MODE 0	MODE 1	MODE 2
PIXELS AFFICHES	160 sur 200	320 sur 200	640 sur 200
NOMBRE PIXELS PAR OCTET	2	4	8
TAILLE DES PIXELS en points	4 sur 2	2 sur 2	1 sur 2
CARACTERES AFFICHES	20 sur 25	40 sur 25	80 sur 25
NOMBRE OCTETS PAR CARACTERE	4	2	1
COULEURS UTILISABLES	16	4	2

Si l'organisation de la mémoire écran n'a plus de secret pour nous, il reste quand même un point essentiel à aborder. Il s'agit de la valeur à attribuer à un octet pour que les pixels qu'il représente soient de la couleur désirée.

Là encore, tout dépend du mode choisi. Commençons par le plus simple, à savoir le mode 0. Dans ce cas, avons-nous vu, un octet représente 8 pixels d'un point chacun. Pour qu'un de ceux-ci soit affiché à l'écran, il suffit de mettre à 1 le bit correspondant et de convertir l'octet ainsi déterminé en décimal ou hexadécimal.

Pour le mode 1, cela se complique. En effet, nous savons qu'un octet sert à coder 4 pixels de 2 points chacun.

Si nous numérotons les bits de l'octet de 7 à 0, de la gauche vers la droite, nous pourrions penser que les bits 7 et 6 servent à fixer l'encre du pixel de l'extrême gauche ; de la même façon les bits 5 et 4 seraient le pixel suivant à droite ; les bits 3 et 2 seraient l'avant-dernier pixel ; et enfin les bits 1 et 0 seraient le pixel d'extrême droite.

Eh bien c'est perdu ! En réalité, le premier pixel à gauche est codé avec les bits 3 et 7, le pixel suivant avec les bits 2 et 6, puis l'avant-dernier avec les bits 1 et 5, et le pixel le plus à droite avec les bits 0 et 4.

Cela nécessite alors d'effectuer de nombreux calculs pour pouvoir colorer chaque pixel avec le numéro d'encre voulu.

Quant au mode 0, le système utilisé ressemble à celui du mode 1. Ici un octet permet de coder 2 pixels de 4 points chacun. Le numéro de l'encre dans laquelle sera affiché le premier pixel est fourni par les bits 1, 5, 3 et 7, et ce dans l'ordre du bit le plus significatif vers le moins significatif. De même, le pixel de droite sera coloré avec le numéro d'encre déterminé par les bits 0, 4, 2 et 6. Là encore, de nombreux calculs seront à faire afin de fournir les couleurs désirées aux pixels.

Enfin, nous ne serions pas complets si nous n'évoquions pas la possibilité de faire afficher les pixels à l'écran selon un cache ; celui-ci permet aux seuls pixels autorisés d'apparaître dans la couleur fixée par le codage de l'octet, et aux autres de s'afficher dans la couleur du fond.

De même les pixels peuvent être affichés dans une autre couleur que celle indiquée par les octets, et ce, en modifiant le mode d'encre graphique comme le précise le guide de l'utilisateur.

COPIE D'UNE ZONE DE L'ÉCRAN

Fonction : copie d'une partie de l'écran à un autre endroit de l'écran.

Il peut être intéressant, notamment dans le cadre d'un logiciel de dessin, de pouvoir recopier une partie de l'écran à un autre endroit de celui-ci. C'est l'objet de la routine suivante.

Le principe en est simple. On fixe, pour la zone à copier, son point de départ, repéré par le coin en bas et à gauche, puis sa longueur et enfin sa hauteur. On précise également, dans les paramètres à fournir à la routine, le nouveau coin en bas et à gauche où se fera la copie de la zone.

A partir de là, chaque ligne de la zone à copier étant une série d'octets, il suffira de déterminer l'adresse de départ de la ligne, puis celle d'arrivée dans la nouvelle zone ; il nous faudra également déterminer le nombre d'octets qu'il faudra transférer par ligne.

Il restera alors, pour chaque ligne, à effectuer le déplacement des octets codant cette ligne.

Syntaxe : |COPZONE,X1,Y1,L,H,X2,Y2

avec X1 : coordonnée X en pixels du coin bas à gauche de la zone à copier.
Y1 : coordonnée Y en pixels du coin bas à gauche de la zone à copier.
L : longueur en pixels de la zone à copier.
H : hauteur en pixels de la zone à copier.
X2 : coordonnée X en pixels du coin bas à gauche de la zone où s'effectue la copie.
Y2 : coordonnée Y en pixels du coin bas à gauche de la zone où s'effectue la copie.

Il sera précisé par la suite pourquoi il est besoin d'exprimer les coordonnées en pixels et non en coordonnées absolues.

Routines système utilisées

Les adresses restent inchangées quel que soit le CPC utilisé.

#BC05 : positionnement du décalage écran.
#BC1D : indique l'adresse d'un pixel dans la mémoire écran.
#BC29 : indique l'adresse de l'octet situé en haut d'un octet dont l'adresse est fournie.

Explications

Les paramètres nécessaires au bon fonctionnement de la routine sont stockés dans DEPX et DEPY pour les coordonnées d'origine de la zone à copier, puis dans COT1 et COT2 pour la longueur et la hauteur de la zone, et enfin dans ARRX et ARRY pour les coordonnées de la zone après copiage.

Pour débuter, la routine prend soin de fixer le décalage écran à 0 afin d'être sûr de n'avoir aucune perturbation à l'écran, comme un scrolling par exemple.

Ensuite se font les recherches des adresses du point d'origine de la première ligne à copier et du point d'arrivée de cette même première ligne. Celles-ci seront conservées dans ADRDEP et ADRARR. Les adresses sont obtenues par appel de la routine #BC1D à laquelle il faut fournir les coordonnées, en pixels, du point dont on désire l'adresse. Cela signifie que ces coordonnées varieront selon le mode dans lequel on travaille. Il faudra alors être vigilant sur ce point afin de ne pas transmettre de coordonnées illégales.

La routine cherche alors le nombre d'octets à copier par ligne. Pour cela, il suffit à nouveau de s'adresser à la routine #BC1D. Nous lui fournirons les coordonnées de la fin de la première ligne. Elles seront obtenues en ajoutant la longueur de la zone (en pixels) à la coordonnée X du coin gauche et en retirant cette adresse à celle de départ de la zone. Le résultat est stocké dans OCLIGN.

Un LDIR se charge ensuite de transférer les octets de la première ligne de ADRDEP à ADRARR, et ce pour le nombre d'octets contenu dans OCLIGN.

COT2 qui représente le nombre de pixels dans la hauteur de la zone à copier est alors décrémentée. En effet, elle fixe aussi le nombre de lignes de la zone et, par là même, quand la décrémentation aboutira à 0, cela signifiera que la dernière aura été copiée.

Tant que la valeur 0 n'est pas atteinte, la routine passe aux calculs des adresses de départ et d'arrivée de la ligne suivante. Ceci se fait par la routine #BC29, qui indique l'adresse, dans la mémoire écran, d'un octet se trouvant juste au-dessus d'un octet dont on lui aura fait parvenir l'adresse.

Une fois ces calculs effectués, les résultats sont stockés dans ADRDEP et ADRARR, afin de pouvoir à nouveau transférer la ligne par un LDIR.

La routine s'achève quand COT2 prend la valeur 0.

Démonstration

La mise en œuvre du programme de démonstration suppose de posséder sur disque ou cassette un fichier, appelé ECRAN.SCR, contenant une page écran en mode 0 ou 1.

Le programme BASIC suivant permet d'effectuer un scrolling d'une bande verticale de la page écran.

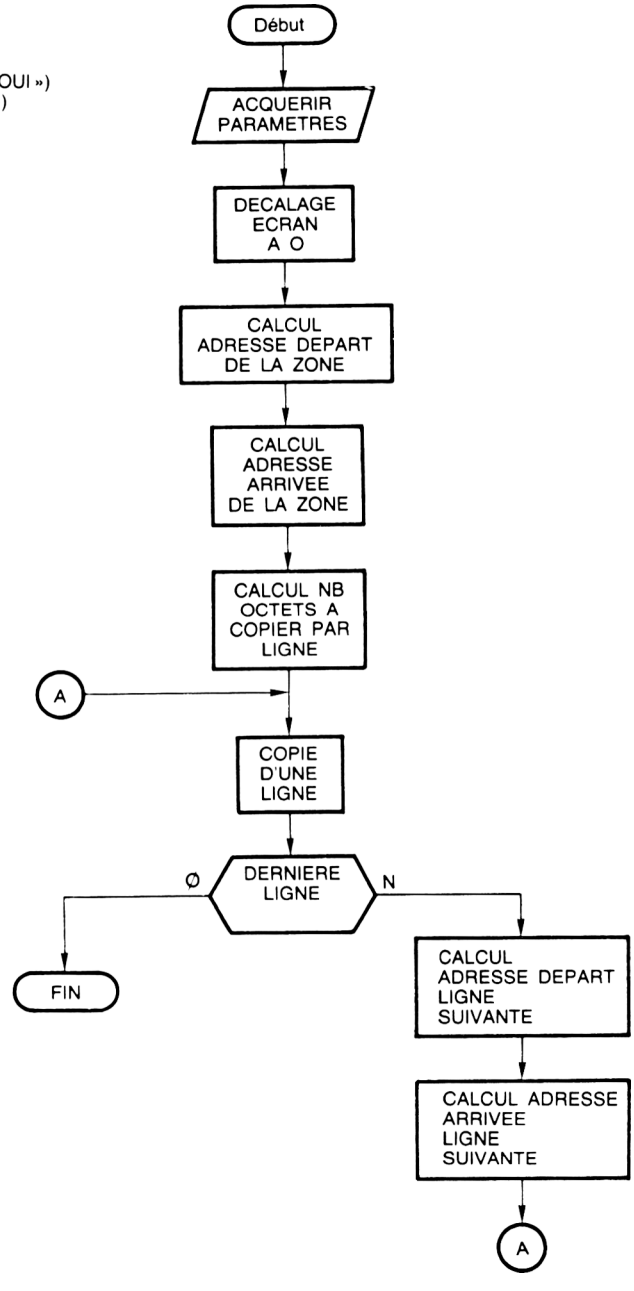
```
10 ink 1,n etc. (Choix des encres avant le changement de la page écran)
20 LOAD « ECRAN.SCR »,&C000
30 FOR I=0 TO 140
40 |COPZONE,100,140-I,60,50,10,10
50 NEXT
```

D'autres effets, que vous aurez tôt fait de découvrir, sont possibles en choisissant bien les paramètres. Faites attention malgré tout à bien fournir les coordonnées en PIXELS. Cependant, l'envoi de coordonnées illégales ne « plante » pas la machine mais permet d'obtenir de curieux résultats.

Routine de copie d'une zone écran à un autre endroit de la page écran

Légende :

0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → (pour « NON »)



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                  10 ;=* ROUTINE COPIE DE ZONE  CPC 464 ET CPC 6128 *=
                  20 ;
6FA0              30          ent  #6fa0
6FA0              40          org  #6fa0
                  50 ;
                  60 ;--* TRAITEMENT RSX *--
                  70 ;
BCD1              80 rsx:     equ  #bcd1
6FA0  01AA6F      90          ld   bc,ext
6FA3  21F56F     100          ld   hl,space
6FA6  CDD1BC     110          call rsx
6FA9  C9         120          ret
6FAA  ED6F       130 ext:     defw nom
                 140 ;--* RECUPERATION DES PARAMETRES *--
6FAC  DD460B     150          ld   b,(ix+11)
6FAF  DD4E0A     160          ld   c,(ix+10)
6FB2  ED436870   170          ld   (depx),bc
6FB6  DD4609     180          ld   b,(ix+9)
6FB9  DD4E08     190          ld   c,(ix+8)
6FBC  ED436A70   200          ld   (depy),bc
6FC0  DD4607     210          ld   b,(ix+7)
6FC3  DD4E06     220          ld   c,(ix+6)
6FC6  ED436C70   230          ld   (cot1),bc
6FCA  DD4605     240          ld   b,(ix+5)
6FCD  DD4E04     250          ld   c,(ix+4)
6FD0  ED436E70   260          ld   (cot2),bc
6FD4  DD4603     270          ld   b,(ix+3)
6FD7  DD4E02     280          ld   c,(ix+2)
6FDA  ED437070   290          ld   (arrx),bc
6FDE  DD4601     300          ld   b,(ix+1)
6FE1  DD4E00     310          ld   c,(ix+0)
6FE4  ED437270   320          ld   (arry),bc
6FE8  FE06       330          cp   6
6FEA  C0         340          ret  nz
6FEB  1813       350          jr   rout
                 360 ;
6FED  434F505A   370 nom:     defm "COPZON"
6FF3  C5         380          defb "E"+#80
6FF4  00         390          defb 0
6FF5             400 space:   defs 4
                 410 ;
                 420 ;
                 430 ;--* TRAITEMENT CALL *--
                 440 ;
```

```
7000                450          org  #7000
                    460 ;
BC1D                470 adcoor: equ  #bc1d
BC05                480 inidec: equ  #bc05
BC29                490 adrhau: equ  #bc29
                    500 ;
                    510 ;--* DECALAGE ECRAN = 0 *--
7000  210000        520 rout:   ld   hl,0
7003  CD05BC        530         call inidec
                    540 ;--* CALCUL ADRESSES DEPART ET ARRIVEE *--
7006  ED5B6870      550         ld   de,(depx)
700A  2A6A70        560         ld   hl,(depy)
700D  CD1DBC        570         call adcoor
7010  227470        580         ld   (adrdep),hl
7013  ED5B7070      590         ld   de,(arrx)
7017  2A7270        600         ld   hl,(arry)
701A  CD1DBC        610         call adcoor
701D  227670        620         ld   (adrarr),hl
                    630 ;--* CALCUL NOMBRE D'OCTETS A COPIER/LIGNE *--
7020  ED4B6870      640         ld   bc,(depx)
7024  2A6C70        650         ld   hl,(cot1)
7027  09            660         add  hl,bc
7028  ED5B6A70      670         ld   de,(depy)
702C  EB            680         ex   de,hl
702D  CD1DBC        690         call adcoor
7030  ED5B7470      700         ld   de,(adrdep)
7034  ED52          710         sbc  hl,de
7036  227870        720         ld   (oclign),hl
                    730 ;--* COPIE D'UNE LIGNE *--
7039  2A7470        740 coplin: ld   hl,(adrdep)
703C  ED5B7670      750         ld   de,(adrarr)
7040  ED4B7870      760         ld   bc,(oclign)
7044  EDB0          770         ldir
                    780 ;--* TEST SI DERNIERE LIGNE A COPIER *--
7046  2A6E70        790         ld   hl,(cot2)
7049  2B            800         dec  hl
704A  226E70        810         ld   (cot2),hl
704D  216E70        820         ld   hl,cot2
7050  3E00          830         ld   a,0
7052  BE            840         cp   (hl)
7053  C8            850         ret  z
                    860 ;--* CALCUL ADRESSES DEP ET ARR LIGNE SUIVANTE *--
7054  2A7470        870         ld   hl,(adrdep)
7057  CD29BC        880         call adrhau
705A  227470        890         ld   (adrdep),hl
705D  2A7670        900         ld   hl,(adrarr)
7060  CD29BC        910         call adrhau
7063  227670        920         ld   (adrarr),hl
7066  18D1          930         jr   coplin
                    940 ;
```

```
7068          950 depx:   defs 2      ;COOR PIX X DEP
706A          960 depy:   defs 2      ;COOR PIX Y DEP
706C          970 cot1:   defs 2      ;NBR PIX LONGUEUR
706E          980 cot2:   defs 2      ;NBR PIX HAUTEUR
7070          990 arrx:   defs 2      ;COOR PIX X ARR
7072         1000 arry:   defs 2      ;COOR PIX Y ARR
7074         1010 adrdep: defs 2      ;STOC ADR DEPART
7076         1020 adrarr: defs 2      ;STOC ADR ARRIVEE
7078         1030 oclign: defs 2      ;STOC NBR OCT/LIG

Pass 2 errors: 00

adcoor BC1D  adrarr 7076  adrdep 7074
adrhau BC29  arrx   7070  arry   7072
coplin 7039  cot1   706C  cot2   706E
depx   7068  depy   706A  ext    6FAA
inidec BC05  nom    6FED  oclign 7078
rout   7000  rsx    BCD1  space  6FF5
```

ZOOM D'UNE PARTIE DE L'ÉCRAN

Fonction : Agrandissement d'une zone de page écran.

Un des principaux attraits d'un logiciel d'aide au dessin est de posséder une fonction ZOOM ou LOUPE permettant d'obtenir un agrandissement d'une zone donnée d'une page écran. Son utilité étant bien entendu de mieux visualiser la zone et surtout de mieux distinguer les couleurs de chaque pixel. C'est l'objet de la routine que nous allons étudier maintenant.

Quel est son principe de fonctionnement ? Il consiste tout simplement à agrandir chaque pixel de la zone concernée à la taille d'un caractère. L'écran pouvant afficher, en mode 0, 25 lignes de 20 caractères et, en mode 1, 25 lignes de 40 caractères, la dimension de la zone à zoomer sera donc, suivant le mode, de 25 lignes de 20 ou 40 pixels. Lors du zoom, chacun de ces pixels sera représenté par un pavé curseur affiché, dans la couleur du pixel qu'il agrandit.

La première tâche de la routine consistera alors à repérer la zone à zoomer grâce aux paramètres qui lui seront fournis, fixant les coordonnées du coin bas et à gauche de la partie à agrandir.

La routine stockera en mémoire les numéros d'encre de chaque pixel de la zone, et ce, ligne à ligne. C'est de cette zone mémoire que le programme récupérera le numéro d'encre pour l'attribuer au stylo lors de l'affichage du zoom.

Ensuite, la page écran devra être sauvegardée soit dans la mémoire, soit sur disque ou cassette, afin de pouvoir être récupérée si besoin.

Il ne restera plus, pour la routine, qu'à afficher, en mode texte, un caractère plein, de la couleur du pixel qu'il agrandit, et ce à une position équivalente à celle occupée par le pixel dans la zone.

L'appui d'une touche permettra de réafficher la page écran telle qu'elle se présentait, avant l'utilisation de cette fonction loupe.

Syntaxe : |ZOOM,X,Y

avec X : coordonnée absolue X du coin bas à gauche de la zone (X compris entre 0 et 639)
Y : coordonnée absolue Y du coin bas à gauche de la zone (Y compris entre 0 et 399).

Routines système utilisées

Entre parenthèses, adresses équivalentes pour le CPC 464.

#B726 (#B285) : pointeur ligne du curseur en mode TEXTE.
#B727 (#B286) : pointeur colonne du curseur en mode TEXTE.

#BB06 : attente de la frappe d'une touche.

#BB5A : impression d'un caractère.
#BB90 : fixe l'encre du stylo.
#BBF0 : indique le numéro de l'encre d'un pixel.
#BC11 : indique le mode en cours.

Explications

Les deux paramètres à passer à la routine (coordonnées absolues du coin bas à gauche de la zone) sont stockés en DEPX et DEPY.

La routine utilisée pour rechercher l'encre d'un pixel se trouve en #BBF0. Elle demande l'envoi des coordonnées absolues d'un point contenu dans ce pixel. Les coordonnées varient donc pour X de 0 à 639, et de 0 à 399 pour Y.

Seulement, selon le mode en cours, les pixels contiennent dans leur longueur, un nombre de points différents : 4 points en mode 0 et 2 points en mode 1. Il faudra alors fixer une valeur que nous appellerons PAS ; celui-ci permettra de déterminer les coordonnées d'un point du pixel suivant par addition de ce PAS à la coordonnée X précédente. PAS contiendra 4 pour le mode 0, et 2 pour le mode 1.

De même, le nombre de colonnes varie selon le mode : 20 pour le mode 0 et 40 pour le mode 1. Cette valeur, qu'il est nécessaire de connaître lors du zoom, sera stockée dans NBRCOL après que la routine, par appel de #BC11, ait déterminé le mode en cours.

En IX sera chargée, puis incrémentée, l'adresse de stockage des numéros d'encre des pixels. Ce stockage s'effectue ligne par ligne de la zone à zoomer. De la première à la dernière colonne de chaque ligne de la zone, la routine cherche le numéro de l'encre du pixel dont les coordonnées sont passées à #BBF0, et la place à l'adresse située dans IX. Le test d'atteinte de la dernière colonne se fait en comparant le registre A, utilisé comme compteur, avec le nombre de colonnes indiqué par NBRCOL.

Quand le compteur arrive à 40 (ou 20 suivant le mode), on passe à la ligne suivante en incrémentant de 2 les coordonnées du début de la ligne précédente car une ligne de pixels fait 2 points de haut. On reprend alors le stockage de la ligne jusqu'à la 25e et dernière ligne de la partie à agrandir.

L'écran est ensuite stocké à l'adresse indiquée dans SAUVEC. Le zoom sera déclenché par l'appui d'une touche, ceci afin d'avoir le temps de visualiser la page écran avant l'exécution de l'effet loupe.

La réalisation du zoom reprend le principe du stockage des numéros d'encre, mais en sens inverse. IX pointe à nouveau sur la zone de stockage des encres. COOX et COOY contiennent les coordonnées de l'endroit où s'affichera le caractère de la couleur du pixel. Ces coordonnées seront placées dans #B725 et #B726 qui sont les pointeurs de la position du curseur texte.

Le numéro de l'encre placé dans l'adresse pointée par IX sert à fixer l'encre du stylo grâce à la routine #BB90. La routine #BB5A affiche alors le caractère de code ASCII #8F qui est en fait, une case pleine. L'affichage s'effectue à la position en cours du curseur. IX et COOX sont incrémentés, la nouvelle encre du stylo fixée, et le caractère #8F affiché à l'écran.

Quand la dernière colonne de la ligne est atteinte, COOY est décrémentée ; en effet l'affichage du zoom se fait du bas vers le haut, donc de la ligne 24 à la ligne 0 et la routine reprend l'affichage de la nouvelle ligne, et ainsi de suite jusqu'à la dernière.

La routine s'achève par le réaffichage de la page écran initiale précédemment stockée. Celui-ci s'effectue après la frappe d'une touche, de manière à pouvoir contrôler le moment de retour de l'écran de départ.

Démonstration

Le programme de démonstration que nous vous proposons demande à nouveau de posséder une page écran en fichier (appelée ECRAN.SCR) comme c'était le cas pour la routine de copie de zone.

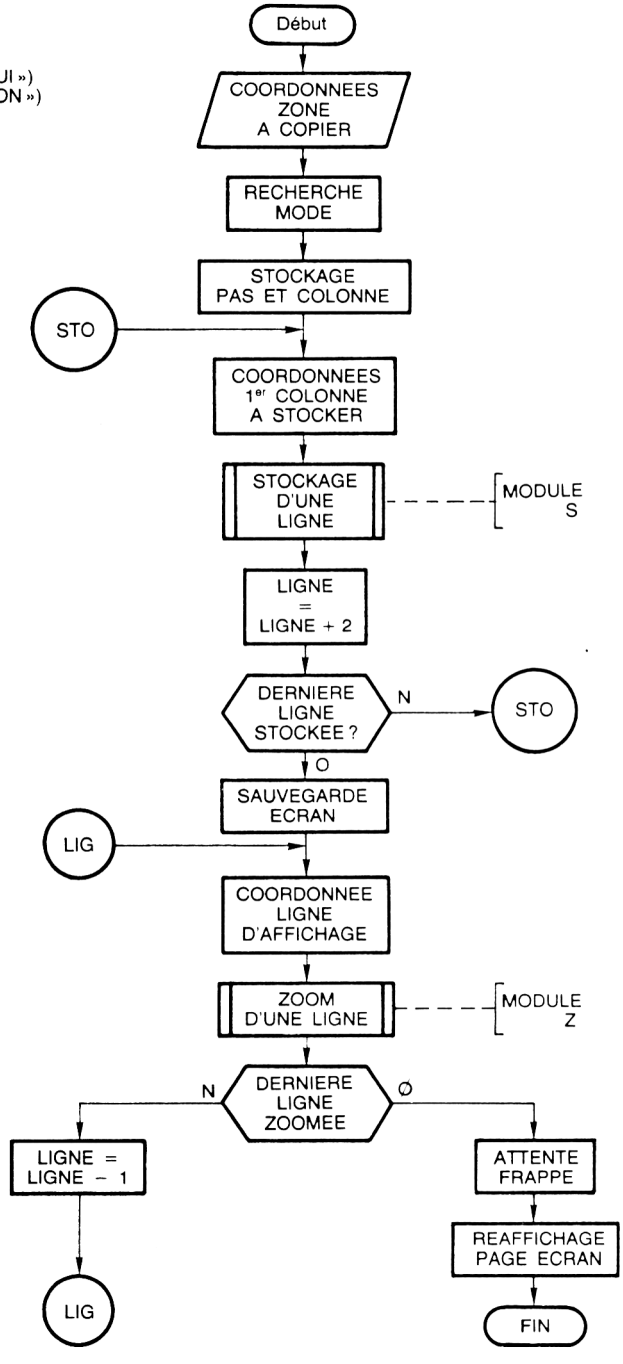
```
10 MODE m (m valant 0 ou 1 selon le cas)
20 INK 1,a: INK 2,b: ... (choix des couleurs des encres)
30 LOAD « ECRAN.SCR »,&C000
60 |ZOOM,320,200
```

Dans ce cas, la partie zoomée sera la zone dont le coin bas à gauche se trouve en plein milieu de l'écran.

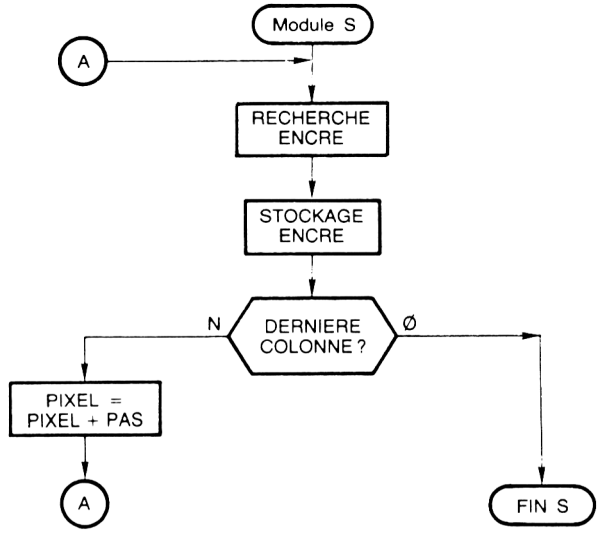
Routine de zoom d'une partie de l'écran graphique

Légende :

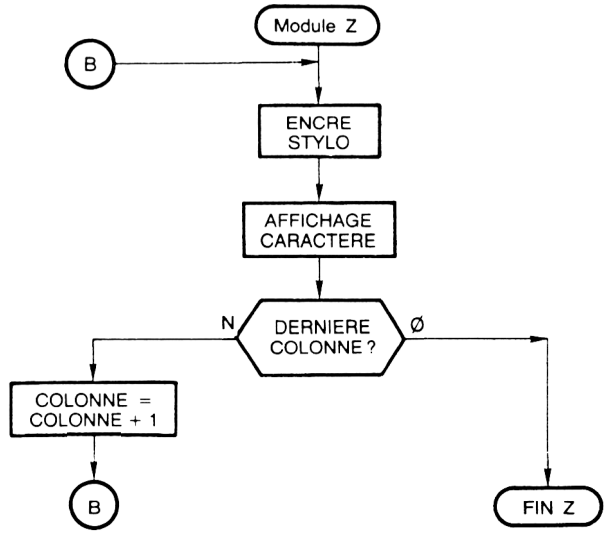
0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → lettre (pour « NON »)



ZOOM : Module de stockage d'une ligne



ZOOM : Module de zoom d'une ligne



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                 10 ;=* ROUTINE ZOOM CPC 6128 *=
                 20 ;
A000             30         ent  #a000
A000             40         org  #a000
                 50 ;
                 60 ;--* TRAITEMENT RSX *--
                 70 ;
BCD1             80 rsx:    equ  #bcd1
A000  010AA0     90         ld   bc,ext
A003  212AA0    100         ld   hl,space
A006  CDD1BC    110         call rsx
A009  C9        120         ret
A00A  25A0      130 ext:    defw nom
                140 ;--* RECUPERATION DES PARAMETRES *--
A00C  DD4601    150         ld   b,(ix+1)
A00F  DD4E00    160         ld   c,(ix+0)
A012  ED432DA1  170         ld   (depy),bc
A016  DD5603    180         ld   d,(ix+3)
A019  DD5E02    190         ld   e,(ix+2)
A01C  ED532BA1  200         ld   (depx),de
A020  FE02      210         cp   2
A022  C0        220         ret  nz
A023  182B      230         jr   rout
                240 ;
A025  5A4F4F    250 nom:    defm "ZOO"
A028  CD        260         defb "M"+#80
A029  00        270         defb 0
A02A            280 space:  defs 4
                290 ;
                300 ;
                310 ;--* TRAITEMENT CALL *--
                320 ;
A050            330         org  #a050
                340 ;
5900            350 sauvec: equ  #5900
BB06            360 fratou: equ  #bb06
BBF0            370 encpix: equ  #bbf0
BB90            380 encsty: equ  #bb90
B727            390 curcol: equ  #b727
B726            400 curlig: equ  #b726
BB5A            410 impcar: equ  #bb5a
BC11            420 mode:   equ  #bc11
                430 ;
                440 ;--* RECHERCHE DU MODE *--
```

```
A050  CD11BC     450 rout:   call mode
A053  FE00       460         cp   0
A055  2810       470         jr   z,a2
                 480 ;--* PAS ET COLONNE POUR MODE 1 *--
A057  010200     490         ld   bc,2
A05A  ED431DA5   500         ld   (pas),bc
A05E  012800     510         ld   bc,40
A061  ED431FA5   520         ld   (nbrcol),bc
A065  180E       530         jr   a3
                 540 ;--* PAS ET COLONNE POUR MODE 0 *--
A067  010400     550 a2:     ld   bc,4
A06A  ED431DA5   560         ld   (pas),bc
A06E  011400     570         ld   bc,20
A071  ED431FA5   580         ld   (nbrcol),bc
                 590 ;--* ADRESSE ZONE DE STOCKAGE DANS IX *--
A075  DD2133A1   600 a3:     ld   ix,stock
                 610 ;--* COORD. 1ere LIGNE A STOCKER DANS HL *--
A079  2A2DA1     620         ld   hl,(depy)
A07C  2231A1     630         ld   (stoy),hl
A07F  3E00       640         ld   a,0
A081  F5         650         push af
                 660 ;--* DEBUT DU STOCKAGE *--
A082  3E00       670 stozon: ld   a,0
                 680 ;--* COORD. COLONNE A STOCKER DANS DE *--
A084  ED5B2BA1   690         ld   de,(depx)
A088  ED532FA1   700         ld   (stox),de
                 710 ;--* STOCKAGE DE LA LIGNE *--
A08C  CDDDA0     720         call stolin
                 730 ;--* LIGNE A STOCKER = LIGNE + 2 *--
A08F  ED5B31A1   740         ld   de,(stoy)
A093  13         750         inc  de
A094  13         760         inc  de
A095  ED5331A1   770         ld   (stoy),de
A099  F1         780         pop  af
A09A  3C         790         inc  a
                 800 ;--* TEST SI LIGNE = 25 *--
A09B  FE19       810         cp   25
                 820 ;--* SI OUI ALLER A SUITE *--
A09D  2803       830         jr   z,suite
A09F  F5         840         push af
A0A0  18E0       850         jr   stozon
                 860 ;--* STOCKAGE DE LA PAGE ECRAN EN SAUVEC *--
A0A2  2100C0     870 suite:  ld   hl,#c000
A0A5  110059     880         ld   de,sauvec
A0A8  010040     890         ld   bc,#4000
A0AB  EDB0       900         ldir
                 910 ;--* ATTENTE FRAPPE D'UNE TOUCHE *--
A0AD  CD06BB     920         call fratou
                 930 ;--* DEBUT DU ZOOM *--
```

```
A0B0  DD2133A1   940 zoomec: ld   ix,stock
                 950 ;--* 1ere LIGNE A ZOOMER EN LIGNE TEXTE 24 *--
A0B4  3E18       960         ld   a,24
A0B6  321CA5     970         ld   (cooy),a
A0B9  3E00       980 a1:     ld   a,0
A0BB  321BA5     990         ld   (coox),a
                1000 ;--* ZOOM D'UNE LIGNE *--
A0BE  CD01A1    1010         call zoomli
A0C1  3A1CA5    1020         ld   a,(cooy)
                1030 ;--* TEST SI DERNIERE LIGNE ? *--
A0C4  FE00      1040         cp   0
                1050 ;--* SI OUI ALLER EN FIN *--
A0C6  2806      1060         jr   z,fin
                1070 ;--* PASSAGE A LA LIGNE TEXTE SUIVANTE *--
A0C8  3D        1080         dec  a
A0C9  321CA5    1090         ld   (cooy),a
A0CC  18EB      1100         jr   a1
                1110 ;--* ATTENTE FRAPPE D'UNE TOUCHE *--
A0CE  CD06BB    1120 fin:    call fratou
                1130 ;--* REAFFICHAGE DE LA PAGE ECRAN *--
A0D1  210059    1140         ld   hl,sauvec
A0D4  1100C0    1150         ld   de,#c000
A0D7  010040    1160         ld   bc,#4000
A0DA  EDB0      1170         ldir
A0DC  C9        1180         ret
                1190 ;
                1200 ;--* STOCKAGE D'UNE LIGNE DE LA ZONE A ZOOMER *--
A0DD  3C        1210 stolin: inc  a
A0DE  F5        1220         push af
                1230 ;--* RECHERCHE DE L'ENCRE DU PIXEL *--
A0DF  ED5B2FA1  1240         ld   de,(stox)
A0E3  2A31A1    1250         ld   hl,(stoy)
A0E6  CDF0BB    1260         call encpix
                1270 ;--* STOCKAGE DE L'ENCRE EN IX *--
A0E9  DD7700    1280         ld   (ix),a
A0EC  DD23      1290         inc  ix
A0EE  F1        1300         pop  af
                1310 ;--* TEST SI DERNIERE COLONNE *--
A0EF  211FA5    1320         ld   hl,nbrcol
A0F2  BE        1330         cp   (hl)
A0F3  C8        1340         ret  z
                1350 ;--* SI NON INCREM. SUIVANT PAS *--
A0F4  2A2FA1    1360         ld   hl,(stox)
A0F7  ED4B1DA5  1370         ld   bc,(pas)
A0FB  09        1380         add  hl,bc
A0FC  222FA1    1390         ld   (stox),hl
A0FF  18DC      1400         jr   stolin
                1410 ;
                1420 ;--* ZOOM D'UNE LIGNE *--
```

```
A101  3C        1430 zoomli: inc  a
A102  F5        1440         push af
                1450 ;--* ENCRE DU STYLO = ENCRE DU PIXEL A ZOOMER *--
A103  DD7E00    1460         ld   a,(ix)
A106  CD90BB    1470         call encsty
A109  DD23      1480         inc  ix
                1490 ;--* AFFICHAGE DU PIXEL ZOOME *--
A10B  3A1BA5    1500         ld   a,(coox)
A10E  3227B7    1510         ld   (curcol),a
A111  3A1CA5    1520         ld   a,(cooy)
A114  3226B7    1530         ld   (curlig),a
A117  3E8F      1540         ld   a,#8f
A119  CD5ABB    1550         call impcar
A11C  F1        1560         pop  af
A11D  211FA5    1570         ld   hl,nbrcol
                1580 ;--* TEST SI DERNIERE COLONNE *--
A120  BE        1590         cp   (hl)
A121  C8        1600         ret  z
                1610 ;--* SI NON COLONNE SUIVANTE *--
A122  3A1BA5    1620         ld   a,(coox)
A125  3C        1630         inc  a
A126  321BA5    1640         ld   (coox),a
A129  18D6      1650         jr   zoomli
                1660 ;
A12B            1670 depx:   defs 2                    ;COOR X ZONE
A12D            1680 depy:   defs 2                    ;COOR Y ZONE
A12F            1690 stox:   defs 2                    ;STOCKAGE DE X
A131            1700 stoy:   defs 2                    ;STOCKAGE DE Y
A133            1710 stock:  defs 1000                 ;ZONE STOCK ENCRE
A51B            1720 coox:   defs 1                    ;STOCK COOR X
A51C            1730 cooy:   defs 1                    ;STOCK COOR Y
A51D            1740 pas:    defs 2                    ;PAS POUR COLONNE
A51F            1750 nbrcol: defs 2                    ;NBRE COL. TEXTE
```

```
Pass 2 errors: 00

al      A0B9  a2      A067  a3      A075
coox    A51B  cooy    A51C  curcol  B727
curlig  B726  depx    A12B  depy    A12D
encpix  BBF0  encsty  BB90  ext     A00A
fin     A0CE  fratou  BB06  impcar  BB5A
mode    BC11  nbrcol  A51F  nom     A025
pas     A51D  rout    A050  rsx     BCD1
sauvec  5900  space   A02A  stock   A133
stolin  A0DD  stox    A12F  stoy    A131
stozon  A082  suite   A0A2  zoomec  A0B0
zoomli  A101

Table used:   370  from   900
Executes: 40960
```

SUPER-ZOOM

Fonction : agrandissement d'une zone de page écran donnée avec affichages simultanés de la zone et de son zoom et possibilité de déplacement.

La routine zoom, vue précédemment, nous a permis de mettre en évidence le principe de l'effet loupe. Mais, faut-il avouer, cela manquait de souplesse dans la mesure où il fallait repasser par la page écran pour voir où se trouvait la partie agrandie.

Nous vous proposons alors une amélioration de la routine zoom, qui permettra d'avoir en même temps à l'écran la partie zoomée, en taille réelle, et, en dessous, le zoom de cette zone. De plus, l'appui sur les touches fléchées autorisera le déplacement du zoom dans la page écran de départ. La touche E permettra de revoir la page écran dans son intégralité sans perdre le zoom et la touche S (stop) signifie l'arrêt de l'utilisation de la routine.

La zone dont on désire l'agrandissement sera placée en haut et au milieu de l'écran, alors que le zoom occupera la majeure partie du reste. Comme le zoom s'affiche en texte, 22 des 25 lignes utilisables lui seront consacrées ; ce qui laisse 3 lignes de texte, soit $3 \times 8 = 24$ lignes de pixels pour la zone en taille réelle. Seules 22 lignes de pixels étant zoomées, il restera une ligne en haut et en bas de la partie non agrandie permettant de la séparer du reste.

De même, selon le mode, 20 ou 40 colonnes sont affichables. Or en mode 0, d'après le tableau vu au début de cette partie, 20 pixels sont codés sur 10 octets et en mode 1, 40 pixels sont aussi codés sur 10 octets. Ce qui signifie que, quel que soit le mode en cours, une ligne de la zone à agrandir représente en mémoire une suite de 10 octets.

La méthode utilisée dans cette routine diffère légèrement de celle employée pour la précédente. En effet, ici, après avoir sauvé en mémoire le contenu de la mémoire écran, il faut rechercher les adresses des octets représentant les lignes de la zone ; puis nous devrons les transférer à la partie de l'écran où s'affiche la zone en taille réelle.

Une fois cette partie reproduite, il suffira, ligne par ligne, puis colonne par colonne, de lire la couleur de chaque pixel ; puis on affichera à la place correspondante dans le zoom un pavé curseur de la couleur du pixel.

Ensuite, il faudra attendre la frappe d'une touche qui modifiera, selon le désir, les coordonnées de la zone à agrandir. La même procédure que pour le premier affichage reprend, et ce jusqu'à l'appui de la touche S.

Syntaxe : |SUPERZOOM,X,Y

avec X : coordonnée X en pixels du coin bas et à gauche de la zone à agrandir.
Y : coordonnée Y en pixels du coin bas et à gauche de la zone à agrandir.

Routines système utilisées

Pour le CPC 464, les adresses équivalentes se trouvent entre parenthèses.

#B726 (#B285) : pointeur ligne du curseur en mode texte.
#B727 (#B286) : pointeur colonne du curseur en mode texte.

#BB00 : initialisation du clavier.
#BB06 : attente de la frappe d'une touche.
#BB5A : impression d'un caractère.
#BB90 : fixe l'encre du stylo en mode texte.
#BBF0 : indique l'encre d'un pixel.
#BC11 : indique le mode en cours.
#BC14 : effacement de l'écran.
#BC1D : indique l'adresse de l'octet d'un pixel.

Explications

La routine débute, après le stockage des coordonnées X et Y dans DEPX et DEPY, par la recherche du mode en cours, par appel de #BC11, pour fixer des paramètres nécessaires à la suite.

Ces paramètres sont :

— le pas (PAS) utilisé pour passer des coordonnées absolues d'un pixel à celles du pixel voisin. PAS = 2 en mode 1 et 4 en mode 0. C'est le nombre de points par pixel,

— le nombre de colonnes (NBRCOL) servira à tester l'arrivée à la dernière colonne pour chaque ligne du zoom, par comparaison avec le contenu du pointeur de colonne. En mode 1, NBRCOL = 39 et en mode 0, NBRCOL = 19 car le pointeur vaut toujours un de moins que la position réelle du curseur,

— COLFEN indique la coordonnée X en pixel du début de la zone de l'écran où se trouve la partie zoomée en taille réelle. COLFEN = 140 en mode 1 et 70 en mode 0.

La routine poursuit par le stockage de la page écran à partir de l'adresse indiquée par SAUVEC. Celle-ci peut être modifiée selon l'emplacement du programme ou des besoins de disponibilité de la mémoire. Il est également possible d'envisager une sauvegarde sur un support magnétique en adaptant la routine dans ce sens.

Le programme principal commence par un CLS d'écran grâce à #BC14. Ensuite a lieu la recopie à l'écran de la zone à zoomer. Pour cela, STO1, STO2, STO3 et STO4 seront des zones de calculs destinées à contenir les coordonnées des débuts de ligne de la partie à agrandir. STO1 et STO2 indiqueront les coordonnées des lignes de la zone dans la page écran initiale ; STO3 et STO4 contiendront, quant à eux, les coordonnées de la même ligne, mais à son emplacement réel à l'écran, là où elle figurera pour le zoom.

Grâce à STO1 et STO2, il est possible de calculer, par #BC1D, l'adresse de stockage du début de chaque ligne de la partie à zoomer ; en effet, on commencera par calculer l'adresse dans la page écran initiale puis, par soustraction de #C000, l'adresse de stockage.

STO3 et STO4 servent à calculer, toujours par #BC1D, l'adresse du début de chaque ligne pour la recopie à l'écran. Une fois ces deux adresses déterminées, un LDIR transfère 10 octets contenant tous les pixels de la ligne. Ceci se repète jusqu'à la copie des 22 lignes de la zone.

Le zoom peut alors commencer. Les zones STO1 et STO2 contiendront les coordonnées absolues des points de la zone à agrandir ; par ailleurs, STO3 et STO4 serviront à stocker les coordonnées du curseur texte placées par la suite dans les pointeurs.

Pour chaque ligne, la couleur de chacun des pixels est fournie par appel de #BBF0, puis donnée au stylo d'écriture de mode texte par #BB90 ; de cette manière on placera dans la case correspondante au pixel un pavé curseur (de code ASCII #8F) par la routine #BB5A. L'opération se reproduit pour les 22 lignes.

Ensuite le clavier est initialisé par #BB00 ; puis par appel de #BB06, la routine attend la frappe d'une touche, dont le code ASCII sera contenu après la frappe dans le registre A. Celui-ci servira, par comparaison, à brancher le programme sur la partie correspondant à la touche enfoncée. L'appui sur les flèches provoque, selon le cas, l'incrémentation ou la décrémentation des coordonnées en pixel du coin de la zone à agrandir, et dans ce cas on retourne en début de routine.

L'appui de la touche E de code ASCII 101 rétablit l'affichage de l'intégralité de la page écran initiale ; la reprise du zoom s'effectuera en frappant une nouvelle flèche. La touche S permet quant à elle de quitter le programme, avec réaffichage auparavant de la page écran de départ.

Démonstration

Là encore, le programme de démonstration utilise la routine sous sa forme d'extension BASIC et nécessite également de posséder un fichier contenant une page écran (appelé ECRAN.SCR).

```
10 MODE m (m = 1 ou 0)
20 INK 1,a: INK 2,b: ... (couleurs des numéros d'encres)
30 LOAD « ECRAN.SCR »,&C000
40 SUPERZOOM,x,y
```

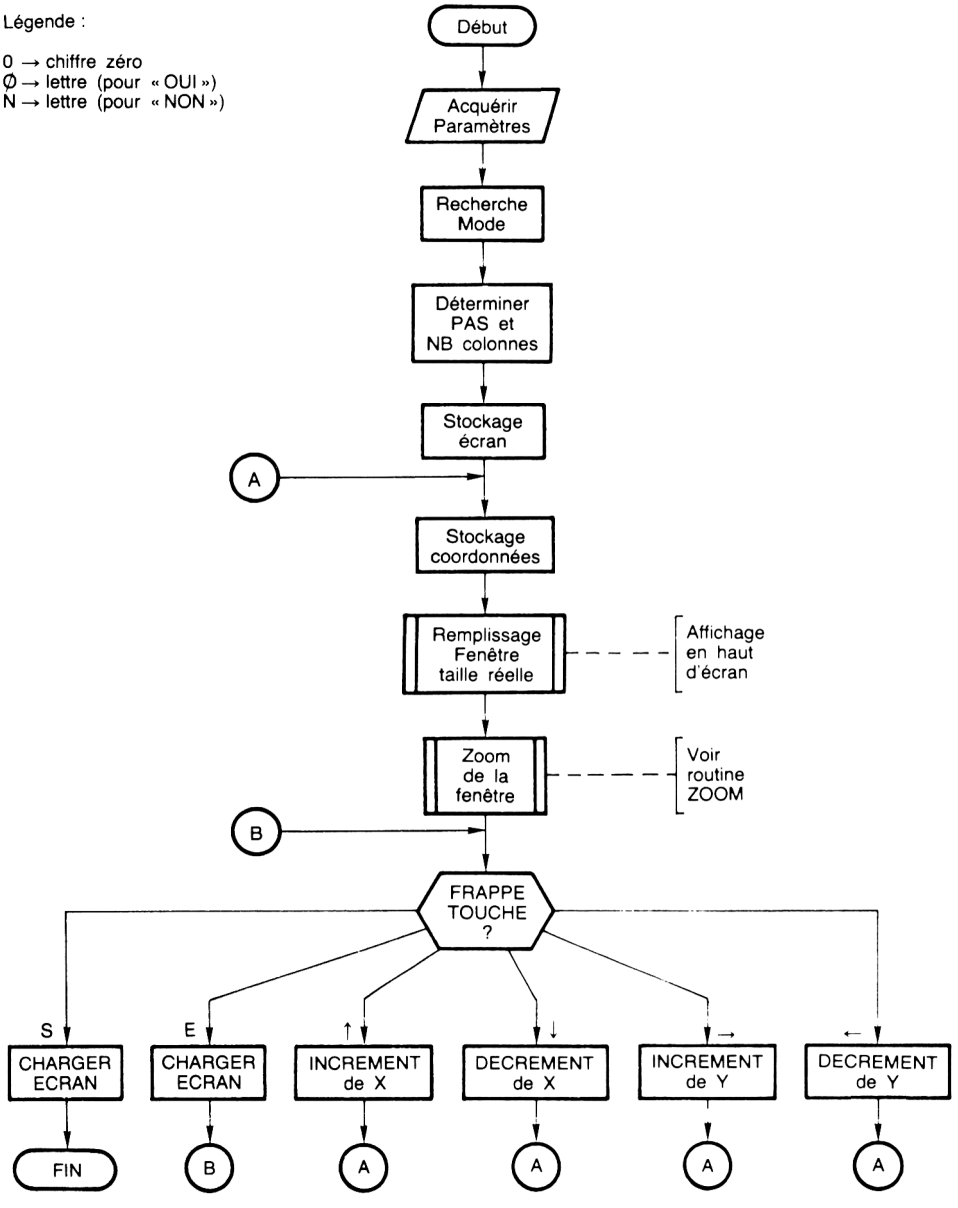
Vous pourrez vérifier que lorsque l'appui sur les touches fléchées gauche ou droite provoque la sortie de la page écran, il est possible d'obtenir un effet de scrolling latéral avec, cependant, parfois un léger décalage. Par contre, un déplacement trop important vers le haut ou vers le bas se traduit quant à lui par un scrolling accompagné généralement par un grand décalage.

Tout ceci s'explique par la gestion mémoire des lignes de pixels de l'écran, gestion abordée en début de chapitre...

Routine « Super- zoom » avec déplacement dans l'écran

Légende :

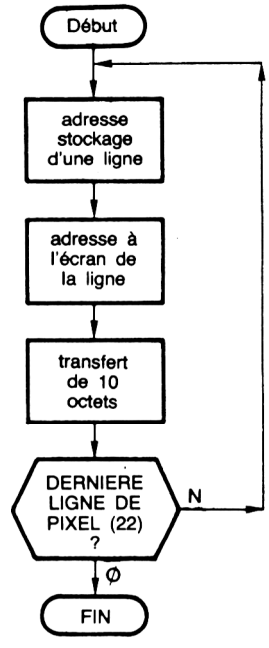
0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → lettre (pour « NON »)



Module de remplissage de la fenêtre en taille réelle

Légende :

0 → chiffre zéro
Ø → lettre (pour « OUI »)
N → lettre (pour « NON »)



Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                 10 ;=* ROUTINE SUPER-ZOOM CPC 6128 *=
                 20 ;
9A00             30          ent  #9a00
9A00             40          org  #9a00
                 50 ;
                 60 ;--* TRAITEMENT RSX *--
                 70 ;
BCD1             80 rsx:     equ  #bcd1
9A00  010A9A     90          ld   bc,ext
9A03  212F9A    100          ld   hl,space
9A06  CDD1BC    110          call rsx
9A09  C9        120          ret
9A0A  259A      130 ext:     defw nom
                140 ;--* RECUPERATION DES PARAMETRES *--
9A0C  DD4601    150          ld   b,(ix+1)
9A0F  DD4E00    160          ld   c,(ix+0)
9A12  ED43B19B  170          ld   (depy),bc
9A16  DD5603    180          ld   d,(ix+3)
9A19  DD5E02    190          ld   e,(ix+2)
9A1C  ED53AF9B  200          ld   (depx),de
9A20  FE02      210          cp   2
9A22  C0        220          ret  nz
9A23  182B      230          jr   rout
                240 ;
9A25  53555045  250 nom:     defm "SUPERZOO"
9A2D  CD        260          defb "M"+#80
9A2E  00        270          defb 0
9A2F            280 space:   defs 4
                290 ;
                300 ;
                310 ;--* TRAITEMENT CALL *--
                320 ;
9A50            330          org  #9a50
                340 ;
5900            350 sauvec: equ  #5900
B726            360 curlig: equ  #b726
B727            370 curcol: equ  #b727
BB00            380 inicla: equ  #bb00
BB06            390 frappe: equ  #bb06
BB5A            400 impcar: equ  #bb5a
BB90            410 encsty: equ  #bb90
BBF0            420 encpix: equ  #bbf0
BC11            430 mode:   equ  #bc11
BC14            440 effecr: equ  #bc14
```

```
BC1D              450 adcoor: equ  #bc1d
                  460 ;
                  470 ;--* RECHERCHE DU MODE *--
9A50  CD11BC      480 rout:   call mode
9A53  FE00        490         cp   0
9A55  2817        500         jr   z,a1
                  510 ;--* PAS ET COLONNE POUR MODE 1 *--
9A57  010200      520         ld   bc,2
9A5A  ED43BD9B    530         ld   (pas),bc
9A5E  012700      540         ld   bc,39
9A61  ED43BF9B    550         ld   (nbrcol),bc
9A65  118C00      560         ld   de,140
9A68  ED53C19B    570         ld   (colfen),de
9A6C  1815        580         jr   stoecr
                  590 ;--* PAS ET COLONNE POUR MODE 0 *--
9A6E  010400      600 a1:     ld   bc,4
9A71  ED43BD9B    610         ld   (pas),bc
9A75  011300      620         ld   bc,19
9A78  ED43BF9B    630         ld   (nbrcol),bc
9A7C  114600      640         ld   de,70
9A7F  ED53C19B    650         ld   (colfen),de
                  660 ;--* STOCKAGE DE LA PAGE ECRAN EN SAUVEC *--
9A83  2100C0      670 stoecr: ld   hl,#c000
9A86  110059      680         ld   de,sauvec
9A89  010040      690         ld   bc,#4000
9A8C  EDB0        700         ldir
                  710 ;--* CLS DE L'ECRAN *--
9A8E  CD14BC      720 remfen: call effecr
                  730 ;--* REMPLISSAGE DE LA FENETRE *--
9A91  ED5BC19B    740         ld   de,(colfen)
9A95  ED53B79B    750         ld   (sto3),de
9A99  21B100      760         ld   hl,177
9A9C  22B99B      770         ld   (sto4),hl
9A9F  ED5BAF9B    780         ld   de,(depx)
9AA3  ED53B39B    790         ld   (sto1),de
9AA7  2AB19B      800         ld   hl,(depy)
9AAA  22B59B      810         ld   (sto2),hl
9AAD  3E01        820         ld   a,1
9AAF  F5          830 a2:     push af
9AB0  ED5BB39B    840         ld   de,(sto1)
9AB4  CD1DBC      850         call adcoor
9AB7  1100C0      860         ld   de,#c000
9ABA  A7          870         and  a
9ABB  ED52        880         sbc  hl,de
9ABD  110059      890         ld   de,sauvec
9AC0  19          900         add  hl,de
9AC1  E5          910         push hl
9AC2  ED5BB79B    920         ld   de,(sto3)
9AC6  2AB99B      930         ld   hl,(sto4)
```

```
9AC9  CD1DBC     940          call adcoor
9ACC  D1         950          pop  de
9ACD  EB         960          ex   de,hl
9ACE  010A00     970          ld   bc,10
9AD1  EDB0       980          ldir
9AD3  2AB99B     990          ld   hl,(sto4)
9AD6  23        1000          inc  hl
9AD7  22B99B    1010          ld   (sto4),hl
9ADA  2AB59B    1020          ld   hl,(sto2)
9ADD  23        1030          inc  hl
9ADE  22B59B    1040          ld   (sto2),hl
9AE1  F1        1050          pop  af
9AE2  3C        1060          inc  a
9AE3  FE17      1070          cp   23
9AE5  20C8      1080          jr   nz,a2
9AE7  1800      1090          jr   zoom
                1100 ;
                1110 ;--* ZOOM DE LA FENETRE *--
9AE9  111800    1120 zoom:    ld   de,24
9AEC  ED53B99B  1130          ld   (sto4),de
9AF0  216201    1140          ld   hl,354
9AF3  1802      1150          jr   a3
9AF5  1897      1160 pont:    jr   remfen
9AF7  22B59B    1170 a3:      ld   (sto2),hl
9AFA  E5        1180          push hl
9AFB  3E00      1190          ld   a,0
9AFD  32B79B    1200          ld   (sto3),a
9B00  111801    1210          ld   de,280
9B03  ED53B39B  1220          ld   (sto1),de
9B07  CD789B    1230          call zoomli
9B0A  21B99B    1240          ld   hl,sto4
9B0D  3E03      1250          ld   a,3
9B0F  BE        1260          cp   (hl)
9B10  2806      1270          jr   z,a4
9B12  35        1280          dec  (hl)
9B13  E1        1290          pop  hl
9B14  23        1300          inc  hl
9B15  23        1310          inc  hl
9B16  18DF      1320          jr   a3
9B18  D1        1330 a4:      pop  de
                1340 ;--* ATTENTE FRAPPE TOUCHE *--
9B19  CD00BB    1350          call inicla
9B1C  CD06BB    1360 touche:  call frappe
9B1F  FE65      1370          cp   101
9B21  CC6C9B    1380          call z,fin
9B24  FE73      1390          cp   115
9B26  2844      1400          jr   z,fin
9B28  FEF3      1410          cp   #f3
9B2A  280E      1420          jr   z,droite
```

```
9B2C  FEF1      1430          cp   #f1
9B2E  2816      1440          jr   z,bas
9B30  FEF2      1450          cp   #f2
9B32  282A      1460          jr   z,gauche
9B34  FEF0      1470          cp   #f0
9B36  281A      1480          jr   z,haut
9B38  18E2      1490          jr   touche
                1500 ;--* VERS LA DROITE *--
9B3A  2AAF9B    1510 droite: ld   hl,(depx)
9B3D  110800    1520          ld   de,8
9B40  19        1530          add  hl,de
9B41  22AF9B    1540          ld   (depx),hl
9B44  18AF      1550          jr   pont
                1560 ;--* VERS LE BAS *--
9B46  2AB19B    1570 bas:    ld   hl,(depy)
9B49  2B        1580          dec  hl
9B4A  2B        1590          dec  hl
9B4B  2B        1600          dec  hl
9B4C  2B        1610          dec  hl
9B4D  22B19B    1620          ld   (depy),hl
9B50  18A3      1630          jr   pont
                1640 ;--* VERS LE HAUT *--
9B52  2AB19B    1650 haut:   ld   hl,(depy)
9B55  23        1660          inc  hl
9B56  23        1670          inc  hl
9B57  23        1680          inc  hl
9B58  23        1690          inc  hl
9B59  22B19B    1700          ld   (depy),hl
9B5C  1897      1710          jr   pont
                1720 ;--* VERS LA GAUCHE *--
9B5E  2AAF9B    1730 gauche: ld   hl,(depx)
9B61  110800    1740          ld   de,8
9B64  A7        1750          and  a
9B65  ED52      1760          sbc  hl,de
9B67  22AF9B    1770          ld   (depx),hl
9B6A  1889      1780          jr   pont
                1790 ;
                1800 ;--* FIN D'UTILISATION *--
9B6C  210059    1810 fin:    ld   hl,sauvec
9B6F  1100C0    1820          ld   de,#c000
9B72  010040    1830          ld   bc,#4000
9B75  EDB0      1840          ldir
9B77  C9        1850          ret
                1860 ;
                1870 ;--* ZOOM D'UNE LIGNE *--
9B78  2AB59B    1880 zoomli: ld   hl,(sto2)
9B7B  ED5BB39B  1890          ld   de,(sto1)
9B7F  CDF0BB    1900          call encpix
9B82  CD90BB    1910          call encsty
```

```
9B85  3AB79B    1920          ld   a,(sto3)
9B88  3227B7    1930          ld   (curcol),a
9B8B  3AB99B    1940          ld   a,(sto4)
9B8E  3226B7    1950          ld   (curlig),a
9B91  3E8F      1960          ld   a,#8f
9B93  CD5ABB    1970          call impcar
9B96  3AB79B    1980          ld   a,(sto3)
9B99  21BF9B    1990          ld   hl,nbrcol
9B9C  BE        2000          cp   (hl)
9B9D  C8        2010          ret  z
9B9E  3C        2020          inc  a
9B9F  32B79B    2030          ld   (sto3),a
9BA2  ED5BB39B  2040          ld   de,(stol)
9BA6  2ABD9B    2050          ld   hl,(pas)
9BA9  19        2060          add  hl,de
9BAA  22B39B    2070          ld   (stol),hl
9BAD  18C9      2080          jr   zoomli
                2090 ;
9BAF            2100 depx:    defs 2                    ;COOR X ZONE
9BB1            2110 depy:    defs 2                    ;COOR Y ZONE
9BB3            2120 stol:    defs 2                    ;ZONE DE CALCUL
9BB5            2130 sto2:    defs 2                    ;ZONE DE CALCUL
9BB7            2140 sto3:    defs 2                    ;ZONE DE CALCUL
9BB9            2150 sto4:    defs 2                    ;ZONE DE CALCUL
9BBB            2160 coox:    defs 1                    ;STOCK COOR X
9BBC            2170 cooy:    defs 1                    ;STOCK COOR Y
9BBD            2180 pas:     defs 2                    ;PAS POUR COLON
9BBF            2190 nbrcol:  defs 2                    ;NBRE COL. TEXTE
9BC1            2200 colfen:  defs 2                    ;COOR X FENETRE

Pass 2 errors: 00

a1      9A6E  a2      9AAF  a3      9AF7
a4      9B18  adcoor  BC1D  bas     9B46
colfen  9BC1  coox    9BBB  cooy    9BBC
curcol  B727  curlig  B726  depx    9BAF
depy    9BB1  droite  9B3A  effecr  BC14
encpix  BBF0  encsty  BB90  ext     9A0A
fin     9B6C  frappe  BB06  gauche  9B5E
haut    9B52  impcar  BB5A  inicla  BB00
mode    BC11  nbrcol  9BBF  nom     9A25
pas     9BBD  pont    9AF5  remfen  9A8E
rout    9A50  rsx     BCD1  sauvec  5900
space   9A2F  stol    9BB3  sto2    9BB5
sto3    9BB7  sto4    9BB9  stoecr  9A83
touche  9B1C  zoom    9AE9  zoomli  9B78

Table used:   498  from   900
Executes: 39424
```

CHAPITRE 5

MATHÉMATIQUES

GÉNÉRALITÉS

L'objet de ce chapitre est de proposer, à l'utilisateur, une série de routines effectuant l'application numérique de formules mathématiques après l'envoi des paramètres adéquats.

Bien sûr, nombre de formules manquent à l'appel. Notre propos étant, non pas de faire une liste exhaustive, mais de fournir un moule pouvant être adapté pour n'importe quel besoin.

Dans ce sens, la programmation a été simplifiée à l'extrême, avec une utilisation maximale des routines mathématiques du système. Là où une programmation plus pointue aurait permis de gagner quelques octets, nous avons choisi la clarté d'un appel à une adresse connue effectuant le même travail.

Cinq routines seront étudiées, traitant des formules d'aire, de volume, d'électricité, de physique et, pour terminer, de quelques calculs comme la résolution d'équations du second degré ou le calcul de somme d'éléments de suites numériques.

Le principe sera, quelle que soit la routine, de la présenter sous forme de RSX. Après le nom de l'extension suivront le numéro de la formule concernée et les adresses des paramètres. Pourquoi ne pas envoyer directement les paramètres comme nous le faisions dans le cas des routines graphiques ? La réponse est simple. On ne peut pas transmettre à un programme en langage machine, par l'intermédiaire d'un CALL, des paramètres dont le codage nécessite plus de 2 octets, donc des paramètres autres qu'entiers.

Dans le programme BASIC, il sera obligatoire de définir les paramètres par une variable. L'adresse de celle-ci est fournie en employant le signe @. Par exemple, si vous désirez appliquer la deuxième formule de la routine PHYSIQUE utilisant 3 paramètres a, b et c, la syntaxe sera : |PHYSIQUE,2,@a,@b,@c, en ayant auparavant attribué à a, b et c leurs valeurs respectives.

La routine débutera toujours par la réception des paramètres. Leurs adresses seront stockées dans les zones STOi et leurs valeurs dans les zones CALCi. Pour reconnaître la formule à utiliser, son numéro sera stocké dans FONCT.

Pour chacune des huit formules, on place le numéro correspondant dans le registre A et on le compare avec le contenu de FONCT jusqu'à ce qu'on obtienne le même. Le nombre de paramètres, mis en pile en début de programme, est alors comparé avec le nombre théorique afin de vérifier si tous ont été fournis. Dans ce cas, la suite de la routine peut s'exécuter.

En règle générale, le résultat des calculs se retrouvera à l'adresse du premier paramètre, grâce à un sous-programme appelé FIN. Dans l'exemple précédent, la réponse sera stockée dans la variable a définie comme étant le premier paramètre.

Routines système utilisées

Entre parenthèses figurent les adresses correspondant au CPC 464, les autres étant celles du CPC 6128.

#BD61 (#BD3D) : copie d'un nombre en FLOTTANT.
#BD64 (#BD40) : conversion ENTIER en FLOTTANT.
#BD6A (#BD46) : conversion FLOTTANT en ENTIER.
#BD79 (#BD55) : multiplication par une puissance de 10.
#BD7C (#BD58) : addition en FLOTTANT.
#BD82 (#BD5B) : soustraction en FLOTTANT.
#BD85 (#BD61) : multiplication en FLOTTANT.
#BD88 (#BD64) : division en FLOTTANT.
#BD8E (#BD6A) : comparaison de 2 nombres en FLOTTANT.
#BD91 (#BD6D) : inversion du signe d'un nombre en FLOTTANT.
#BD94 (#BD70) : test du signe d'un nombre en FLOTTANT.
#BD97 (#BD73) : choix de l'unité d'angle.
#BD9A (#BD76) : détermination de PI.
#BD9D (#BD79) : racine d'un nombre en FLOTTANT.
#BDA0 (#BD7C) : élévation d'un nombre à la puissance.
#BDAC (#BD88) : calcul du sinus d'un angle.
#BDAF (#BD8B) : calcul du cosinus d'un angle.

AIRES

NUMERO FONCTION	FIGURE	NOMBRE PARAM.	PARAMETRES	FORMULES
1	TRIANGLE	2	B base h hauteur	B×h/2
2	LOSANGE	2	a grd diag b pte diag	a×b/2
3	TRAPEZE	3	a grd base b pte base h hauteur	(a+b)×h/2
4	DISQUE	1	R rayon	R^2×pi
5	SPHERE	1	R rayon	4×pi×R^2
6	ELLIPSE	2	a axe1/2 b axe2/2	a×b×pi
7	PAROI DE CONE	2	R ray base A générat.	R×A×pi
8	PAROI DE CYLINDRE	2	R ray base h hauteur	2×pi×R×h

Aire d'un triangle

|AIRE,1,@B,@H A = B×h/2

En entrée : B base du triangle.
h hauteur associée à la base.

En sortie : B aire du triangle.

L'aire du triangle se retrouvera, après exécution de la routine, à l'adresse contenant B lors de la création des variables.

Aire d'un losange

|AIRE,2,@a,@b $A = a \times b/2$

En entrée : a grande diagonale.
b petite diagonale.

En sortie : a aire du losange.

Noter que la formule étant identique à celle du triangle, c'est celle-ci qui est utilisée pour effectuer le calcul d'aire.

Aire d'un trapèze

|AIRE,3,@a,@b,@h $A = (a+b) \times h/2$

En entrée : a petite base.
b grande base.
h hauteur.

En sortie : a aire du trapèze.

Aire d'un disque

|AIRE,4,@R $A = 2 \times pi \times R$

En entrée: a rayon du cercle.

En sortie : R aire du disque.

Aire d'une sphère

|AIRE,5,@R $A = 4 \times pi \times R\hat{}2$

En entrée : R rayon de la sphère.

En sortie : R aire de la sphère.

L'aire d'une sphère correspond bien entendu à l'aire de la paroi de la sphère.

Aire d'une ellipse

|AIRE,6,@a,@b $A = a \times b \times pi$

En entrée : a premier demi-axe.
b deuxième demi-axe.

En sortie : a aire de l'ellipse.

Aire de la paroi d'un cône

|AIRE,7,@R,@A $A = R \times A \times pi$

En entrée : R rayon de la base.
A génératrice du cône.

En sortie : R aire de la paroi.

Aire de la paroi d'un cylindre

|AIRE,8,@R,@h $A = 2 \times pi \times R \times h$

En entrée : R rayon de la base.
A hauteur du cylindre.

En sortie : a aire de la paroi.

La paroi d'un cylindre est en fait, une fois étalée, un rectangle dont la longueur représente le périmètre du cercle de base, et la largeur la hauteur du cylindre.

```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                  10 ;=*  ROUTINE AIRE CPC 6128  *=
                  20 ;
8000              30          ent  #8000
8000              40          org  #8000
                  50 ;
                  60 ;--* TRAITEMENT RSX *--
                  70 ;
BCD1              80 rsx:     equ  #bcd1
8000  010A80      90          ld   bc,ext
8003  216280     100          ld   hl,space
8006  CDD1BC     110          call rsx
8009  C9         120          ret
800A  5D80       130 ext:     defw nom
                 140 ;--* RECUPERATION DES PARAMETRES *--
                 150 ;--* TEST SI 4 PARAMETRES *--
800C  F5         160          push af
800D  FE04       170          cp   4
800F  2014       180          jr   nz,a1
8011  DD6601     190          ld   h,(ix+1)
8014  DD6E00     200          ld   l,(ix+0)
8017  22C881     210          ld   (sto3),hl
801A  11D481     220          ld   de,calc3
801D  CDB181     230          call stock
8020  DD23       240          inc  ix
8022  DD23       250          inc  ix
8024  3D         260          dec  a
                 270 ;--* TEST SI 3 PARAMETRES *--
8025  FE03       280 a1:      cp   3
8027  2014       290          jr   nz,a2
8029  DD6601     300          ld   h,(ix+1)
802C  DD6E00     310          ld   l,(ix+0)
802F  22C681     320          ld   (sto2),hl
```

```
8032  11CF81    330          ld   de,calc2
8035  CDB181    340          call stock
8038  DD23      350          inc  ix
803A  DD23      360          inc  ix
803C  3D        370          dec  a
                380 ;--* TEST SI 2 PARAMETRES *--
803D  FE02      390 a2:      cp   2
803F  201A      400          jr   nz,a3
8041  DD6601    410          ld   h,(ix+1)
8044  DD6E00    420          ld   l,(ix+0)
8047  22C481    430          ld   (stol),hl
804A  11CA81    440          ld   de,calcl
804D  CDB181    450          call stock
8050  DD6603    460          ld   h,(ix+3)
8053  DD6E02    470          ld   l,(ix+2)
8056  22DE81    480          ld   (fonct),hl
8059  1825      490          jr   rout
805B  F1        500 a3:      pop  af
805C  C9        510          ret
                520 ;
805D  414952    530 nom:     defm "AIR"
8060  C5        540          defb "E"+#80
8061  00        550          defb 0
8062            560 space:   defs 4
                570 ;
                580 ;
                590 ;--* TRAITEMENT CALL *--
                600 ;
8080            610          org  #8080
                620 ;
BD64            630 entflo:  equ  #bd64
BD7C            640 additi:  equ  #bd7c
BD85            650 multip:  equ  #bd85
BD88            660 divise:  equ  #bd88
BD9A            670 pi:      equ  #bd9a
                680 ;
                690 ;--* AIRE D'UN TRIANGLE *--
8080  3E01      700 rout:    ld   a,l
8082  21DE81    710          ld   hl,fonct
8085  BE        720          cp   (hl)
8086  2024      730          jr   nz,losang
8088  F1        740 a4:      pop  af
8089  FE03      750          cp   3
808B  C0        760          ret  nz
808C  21CA81    770          ld   hl,calcl
808F  11CF81    780          ld   de,calc2
8092  CD85BD    790          call multip
8095  AF        800          xor  a
8096  210200    810          ld   hl,2
```

```
8099  11D481    820          ld   de,calc3
809C  CD64BD    830          call entflo
809F  21CA81    840          ld   hl,calcl
80A2  11D481    850          ld   de,calc3
80A5  CD88BD    860          call divise
80A8  CDB781    870          call fin
80AB  C9        880          ret
                890 ;--* AIRE D'UN LOSANGE *--
80AC  3E02      900 losang: ld   a,2
80AE  21DE81    910          ld   hl,fonct
80B1  BE        920          cp   (hl)
80B2  2002      930          jr   nz,trapez
80B4  18D2      940          jr   a4
                950 ;--* AIRE D'UN TRAPEZE *--
80B6  3E03      960 trapez: ld   a,3
80B8  21DE81    970          ld   hl,fonct
80BB  BE        980          cp   (hl)
80BC  2027      990          jr   nz,cercle
80BE  F1       1000          pop  af
80BF  FE04     1010          cp   4
80C1  C0       1020          ret  nz
80C2  AF       1030          xor  a
80C3  210200   1040          ld   hl,2
80C6  11D981   1050          ld   de,calc4
80C9  CD64BD   1060          call entflo
80CC  21CA81   1070          ld   hl,calcl
80CF  11CF81   1080          ld   de,calc2
80D2  CD7CBD   1090          call additi
80D5  11D481   1100          ld   de,calc3
80D8  CD85BD   1110          call multip
80DB  11D981   1120          ld   de,calc4
80DE  CD88BD   1130          call divise
80E1  CDB781   1140          call fin
80E4  C9       1150          ret
               1160 ;--* AIRE D'UN CERCLE *--
80E5  3E04     1170 cercle: ld   a,4
80E7  21DE81   1180          ld   hl,fonct
80EA  BE       1190          cp   (hl)
80EB  201D     1200          jr   nz,sphere
80ED  F1       1210          pop  af
80EE  FE02     1220          cp   2
80F0  C0       1230          ret  nz
80F1  21CF81   1240          ld   hl,calc2
80F4  CD9ABD   1250          call pi
80F7  21CA81   1260          ld   hl,calcl
80FA  11CA81   1270          ld   de,calcl
80FD  CD85BD   1280          call multip
8100  11CF81   1290          ld   de,calc2
8103  CD85BD   1300          call multip
```

```
8106  CDB781   1310          call fin
8109  C9       1320          ret
               1330 ;--* AIRE D'UNE SPHERE *--
810A  3E05     1340 sphere: ld   a,5
810C  21DE81   1350          ld   hl,fonct
810F  BE       1360          cp   (hl)
8110  202D     1370          jr   nz,ellips
8112  F1       1380          pop  af
8113  FE02     1390          cp   2
8115  C0       1400          ret  nz
8116  21CF81   1410          ld   hl,calc2
8119  CD9ABD   1420          call pi
811C  AF       1430          xor  a
811D  210400   1440          ld   hl,4
8120  11D481   1450          ld   de,calc3
8123  CD64BD   1460          call entflo
8126  21CA81   1470          ld   hl,calc1
8129  11CA81   1480          ld   de,calc1
812C  CD85BD   1490          call multip
812F  11CF81   1500          ld   de,calc2
8132  CD85BD   1510          call multip
8135  11D481   1520          ld   de,calc3
8138  CD85BD   1530          call multip
813B  CDB781   1540          call fin
813E  C9       1550          ret
               1560 ;--* AIRE D'UNE ELLIPSE *--
813F  3E06     1570 ellips: ld   a,6
8141  21DE81   1580          ld   hl,fonct
8144  BE       1590          cp   (hl)
8145  201D     1600          jr   nz,cone
8147  F1       1610          pop  af
8148  FE03     1620          cp   3
814A  C0       1630          ret  nz
814B  21D481   1640          ld   hl,calc3
814E  CD9ABD   1650          call pi
8151  21CA81   1660          ld   hl,calc1
8154  11CF81   1670          ld   de,calc2
8157  CD85BD   1680          call multip
815A  11D481   1690          ld   de,calc3
815D  CD85BD   1700          call multip
8160  CDB781   1710          call fin
8163  C9       1720          ret
               1730 ;--* AIRE DE LA PAROI D'UN CONE *--
8164  3E07     1740 cone:   ld   a,7
8166  21DE81   1750          ld   hl,fonct
8169  BE       1760          cp   (hl)
816A  201D     1770          jr   nz,cylind
816C  F1       1780          pop  af
816D  FE03     1790          cp   3
```

```
816F  C0        1800          ret  nz
8170  21D481    1810          ld   hl,calc3
8173  CD9ABD    1820          call pi
8176  21CA81    1830          ld   hl,calc1
8179  11CF81    1840          ld   de,calc2
817C  CD85BD    1850          call multip
817F  11D481    1860          ld   de,calc3
8182  CD85BD    1870          call multip
8185  CDB781    1880          call fin
8188  C9        1890          ret
                1900 ;--* AIRE DE LA PAROI D'UN CYLINDRE *--
8189  3E08      1910 cylind: ld   a,8
818B  21DE81    1920          ld   hl,fonct
818E  BE        1930          cp   (hl)
818F  C0        1940          ret  nz
8190  F1        1950          pop  af
8191  FE03      1960          cp   3
8193  C0        1970          ret  nz
8194  21D481    1980          ld   hl,calc3
8197  CD9ABD    1990          call pi
819A  11D481    2000          ld   de,calc3
819D  CD7CBD    2010          call additi
81A0  11CA81    2020          ld   de,calc1
81A3  EB        2030          ex   de,hl
81A4  CD85BD    2040          call multip
81A7  11CF81    2050          ld   de,calc2
81AA  CD85BD    2060          call multip
81AD  CDB781    2070          call fin
81B0  C9        2080          ret
                2090 ;
81B1  010500    2100 stock:  ld   bc,5
81B4  EDB0      2110          ldir
81B6  C9        2120          ret
81B7  21CA81    2130 fin:    ld   hl,calc1
81BA  ED5BC481  2140          ld   de,(sto1)
81BE  010500    2150          ld   bc,5
81C1  EDB0      2160          ldir
81C3  C9        2170          ret
                2180 ;
81C4            2190 sto1:   defs 2
81C6            2200 sto2:   defs 2
81C8            2210 sto3:   defs 2
81CA            2220 calc1:  defs 5
81CF            2230 calc2:  defs 5
81D4            2240 calc3:  defs 5
81D9            2250 calc4:  defs 5
81DE            2260 fonct:  defs 2
```

```
Pass 2 errors: 00

a1      8025  a2      803D  a3      805B
a4      8088  additi  BD7C  calc1   81CA
calc2   81CF  calc3   81D4  calc4   81D9
cercle  80E5  cone    8164  cylind  8189
divise  BD88  ellips  813F  entflo  BD64
ext     800A  fin     81B7  fonct   81DE
losang  80AC  multip  BD85  nom     805D
pi      BD9A  rout    8080  rsx     BCD1
space   8062  sphere  810A  sto1    81C4
sto2    81C6  sto3    81C8  stock   81B1
trapez  80B6

Table used:   367  from   900
Executes: 32768
```

VOLUMES

NUMERO FONCTION	FIGURE	NOMBRE PARAM.	PARAMETRES	FORMULES
1	CUBE	1	A côté	A^3
2	SPHERE	1	R rayon	4×pi×R^3/3
3	PRISME DROIT	2	B aire base h hauteur	B×h
4	CONE DROIT	2	R ray base h hauteur	pi×h×R^2/3
5	CONE TRONQUE	3	a ray BASE b ray base h hauteur	pi×h(a^2+ab+b^2) ——— 3
6	PARALLELEPI.	3	a longueur b largeur h hauteur	a×b×h
7	PYRAMIDE	2	B base h hauteur	B×h/3
8	CYLINDRE	2	R ray base h hauteur	pi×h×R^2

Volume d'un cube

|VOLUME,1,@A V = A^3

En entrée : A arête du cube.

En sortie : A volume du cube.

Volume d'une sphère

|VOLUME,2,@R V = 4×pi×R^3/3

En entrée : R rayon de la sphère.

En sortie : R volume de la sphère.

Volume d'un prisme droit

|VOLUME,3,@B,@h $V = B \times h$

En entrée : B aire de la base.
h hauteur du prisme.

En sortie : B volume du prisme.

Un prisme droit est un volume formé d'une base plane quelconque, d'aire B et de bords droits, c'est-à-dire perpendiculaires par rapport à la surface de base. La surface fermant le prisme est bien sûr identique à celle de base et lui est parallèle.

Volume d'un cône droit

|VOLUME,4,@R,@h $V = pi \times R\hat{}2 \times h/3$

En entrée : R rayon de la base.
h hauteur du cône.

En sortie : R volume du cône.

La formule permettant de calculer le volume d'un cône penché est identique à celle-ci. Mais dans ce cas, la hauteur ne représente plus la distance entre le sommet du cône et le centre du disque de base, mais la distance entre le sommet du cône et le plan contenant le disque de base.

Volume d'un cône tronqué

|VOLUME,5,@a,@b,@h $V = pi \times h \times (a\hat{}2 + a \times b + b\hat{}2)/3$

En entrée : a rayon grande base.
b rayon petite base
h hauteur du cône.

En sortie : a volume du cône tronqué.

Un cône tronqué est un cône étêté. C'est donc un cône normal auquel on aurait retiré au sommet un cône de rayon de base b.

A noter que dans la routine, les paramètres a et b étant utilisés deux fois, ils sont alors, en début d'exécution, recopiés dans CALC6 et CALC7.

Volume d'un parallélépipède

|VOLUME,6,@a,@b,@h $V = a \times b \times h$

En entrée : a longueur de la base.
b largeur de la base.
h hauteur.

En sortie : a volume du parallélépipède.

Le volume d'un parallélépipède dont les bords verticaux ne sont plus droits par rapport à la surface de base est obtenu par la même formule. h représente alors la distance entre les deux surfaces horizontales.

Volume d'une pyramide

|VOLUME,7,@B,@h V = B×h/3

En entrée : B aire de la base.
h hauteur de la pyramide.

En sortie : B volume de la pyramide.

Volume d'un cylindre

|VOLUME,8,@R,@h V = pi×R^2×h

En entrée : R rayon de la base.
h hauteur du cylindre.

En sortie : R volume du cylindre.

Le volume du cylindre est obtenu par la multiplication de l'aire de base par la hauteur. La surface de base étant un disque de rayon R, son aire est pi×R^2.

```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                  10 ;=*  ROUTINE VOLUME CPC 6128  *=
                  20 ;
8000              30          ent  #8000
8000              40          org  #8000
                  50 ;
                  60 ;--* TRAITEMENT RSX *--
                  70 ;
BCD1              80 rsx:     equ  #bcd1
8000  010A80      90          ld   bc,ext
8003  216480     100          ld   hl,space
8006  CDD1BC     110          call rsx
8009  C9         120          ret
800A  5D80       130 ext:     defw nom
                 140 ;--* RECUPERATION DES PARAMETRES *--
                 150 ;--* TEST SI 4 PARAMETRES *--
800C  F5         160          push af
800D  FE04       170          cp   4
800F  2014       180          jr   nz,al
8011  DD6601     190          ld   h,(ix+1)
8014  DD6E00     200          ld   l,(ix+0)
8017  223482     210          ld   (sto3),hl
801A  114082     220          ld   de,calc3
801D  CD1D82     230          call stock
8020  DD23       240          inc  ix
```

```
8022  DD23       250          inc  ix
8024  3D         260          dec  a
                 270 ;--* TEST SI 3 PARAMETRES *--
8025  FE03       280 a1:      cp   3
8027  2014       290          jr   nz,a2
8029  DD6601     300          ld   h,(ix+1)
802C  DD6E00     310          ld   l,(ix+0)
802F  223282     320          ld   (sto2),hl
8032  113B82     330          ld   de,calc2
8035  CD1D82     340          call stock
8038  DD23       350          inc  ix
803A  DD23       360          inc  ix
803C  3D         370          dec  a
                 380 ;--* TEST SI 2 PARAMETRES *--
803D  FE02       390 a2:      cp   2
803F  201A       400          jr   nz,a3
8041  DD6601     410          ld   h,(ix+1)
8044  DD6E00     420          ld   l,(ix+0)
8047  223082     430          ld   (sto1),hl
804A  113682     440          ld   de,calc1
804D  CD1D82     450          call stock
8050  DD6603     460          ld   h,(ix+3)
8053  DD6E02     470          ld   l,(ix+2)
8056  225982     480          ld   (fonct),hl
8059  1825       490          jr   rout
805B  F1         500 a3:      pop  af
805C  C9         510          ret
                 520 ;
805D  564F4C55   530 nom:     defm "VOLUM"
8062  C5         540          defb "E"+#80
8063  00         550          defb 0
8064             560 space:   defs 4
                 570 ;
                 580 ;
                 590 ;--* TRAITEMENT CALL *--
                 600 ;
8080             610          org  #8080
                 620 ;
BD61             630 copie:   equ  #bd61
BD64             640 entflo:  equ  #bd64
BD7C             650 additi:  equ  #bd7c
BD85             660 multip:  equ  #bd85
BD88             670 divise:  equ  #bd88
BD9A             680 pi:      equ  #bd9a
BDA0             690 puiss:   equ  #bda0
                 700 ;
                 710 ;--* VOLUME D'UN CUBE *--
8080  3E01       720 rout:    ld   a,1
8082  215982     730          ld   hl,fonct
```

```
8085  BE        740          cp   (hl)
8086  2020      750          jr   nz,sphere
8088  F1        760 cube:    pop  af
8089  FE02      770          cp   2
808B  C0        780          ret  nz
808C  113682    790          ld   de,calc1
808F  213B82    800          ld   hl,calc2
8092  CD61BD    810          call copie
8095  213682    820          ld   hl,calc1
8098  113682    830          ld   de,calc1
809B  CD85BD    840          call multip
809E  113B82    850          ld   de,calc2
80A1  CD85BD    860          call multip
80A4  CD2382    870          call fin
80A7  C9        880          ret
                890 ;--* VOLUME D'UNE SPHERE *--
80A8  3E02      900 sphere:  ld   a,2
80AA  215982    910          ld   hl,fonct
80AD  BE        920          cp   (hl)
80AE  203D      930          jr   nz,prisme
80B0  F1        940          pop  af
80B1  FE02      950          cp   2
80B3  C0        960          ret  nz
80B4  AF        970          xor  a
80B5  210400    980          ld   hl,4
80B8  113B82    990          ld   de,calc2
80BB  CD64BD   1000          call entflo
80BE  AF       1010          xor  a
80BF  210300   1020          ld   hl,3
80C2  114082   1030          ld   de,calc3
80C5  CD64BD   1040          call entflo
80C8  214582   1050          ld   hl,calc4
80CB  CD9ABD   1060          call pi
80CE  213682   1070          ld   hl,calc1
80D1  114082   1080          ld   de,calc3
80D4  CDA0BD   1090          call puiss
80D7  113B82   1100          ld   de,calc2
80DA  CD85BD   1110          call multip
80DD  114582   1120          ld   de,calc4
80E0  CD85BD   1130          call multip
80E3  114082   1140          ld   de,calc3
80E6  CD88BD   1150          call divise
80E9  CD2382   1160          call fin
80EC  C9       1170          ret
               1180 ;--* VOLUME D'UN PRISME *--
80ED  3E03     1190 prisme:  ld   a,3
80EF  215982   1200          ld   hl,fonct
80F2  BE       1210          cp   (hl)
80F3  2011     1220          jr   nz,conedt
```

```
80F5  F1        1230          pop  af
80F6  FE03      1240          cp   3
80F8  C0        1250          ret  nz
80F9  213682    1260          ld   hl,calc1
80FC  113B82    1270          ld   de,calc2
80FF  CD85BD    1280          call multip
8102  CD2382    1290          call fin
8105  C9        1300          ret
                1310 ;--* VOLUME D'UN CONE DROIT *--
8106  3E04      1320 conedt: ld   a,4
8108  215982    1330          ld   hl,fonct
810B  BE        1340          cp   (hl)
810C  2032      1350          jr   nz,conetq
810E  F1        1360          pop  af
810F  FE03      1370          cp   3
8111  C0        1380          ret  nz
8112  AF        1390          xor  a
8113  210300    1400          ld   hl,3
8116  114582    1410          ld   de,calc4
8119  CD64BD    1420          call entflo
811C  214082    1430          ld   hl,calc3
811F  CD9ABD    1440          call pi
8122  213682    1450          ld   hl,calc1
8125  113682    1460          ld   de,calc1
8128  CD85BD    1470          call multip
812B  113B82    1480          ld   de,calc2
812E  CD85BD    1490          call multip
8131  114082    1500          ld   de,calc3
8134  CD85BD    1510          call multip
8137  114582    1520          ld   de,calc4
813A  CD88BD    1530          call divise
813D  CD2382    1540          call fin
                1550 ;--* VOLUME D'UN CONE TRONQUE *--
8140  3E05      1560 conetq: ld   a,5
8142  215982    1570          ld   hl,fonct
8145  BE        1580          cp   (hl)
8146  2063      1590          jr   nz,parall
8148  F1        1600          pop  af
8149  FE04      1610          cp   4
814B  C0        1620          ret  nz
814C  214582    1630          ld   hl,calc4
814F  CD9ABD    1640          call pi
8152  AF        1650          xor  a
8153  210300    1660          ld   hl,3
8156  114A82    1670          ld   de,calc5
8159  CD64BD    1680          call entflo
815C  113682    1690          ld   de,calc1
815F  214F82    1700          ld   hl,calc6
8162  CD61BD    1710          call copie
```

```
8165  113B82   1720          ld   de,calc2
8168  215482   1730          ld   hl,calc7
816B  CD61BD   1740          call copie
816E  214F82   1750          ld   hl,calc6
8171  114F82   1760          ld   de,calc6
8174  CD85BD   1770          call multip
8177  215482   1780          ld   hl,calc7
817A  115482   1790          ld   de,calc7
817D  CD85BD   1800          call multip
8180  213682   1810          ld   hl,calc1
8183  113B82   1820          ld   de,calc2
8186  CD85BD   1830          call multip
8189  114F82   1840          ld   de,calc6
818C  CD7CBD   1850          call additi
818F  115482   1860          ld   de,calc7
8192  CD7CBD   1870          call additi
8195  114082   1880          ld   de,calc3
8198  CD85BD   1890          call multip
819B  114582   1900          ld   de,calc4
819E  CD85BD   1910          call multip
81A1  114A82   1920          ld   de,calc5
81A4  CD88BD   1930          call divise
81A7  CD2382   1940          call fin
81AA  C9       1950          ret
               1960 ;--* VOLUME D'UN PARALLELEPIPEDE *--
81AB  3E06     1970 parall: ld   a,6
81AD  215982   1980          ld   hl,fonct
81B0  BE       1990          cp   (hl)
81B1  2017     2000          jr   nz,pyrami
81B3  F1       2010          pop  af
81B4  FE04     2020          cp   4
81B6  C0       2030          ret  nz
81B7  213682   2040          ld   hl,calc1
81BA  113B82   2050          ld   de,calc2
81BD  CD85BD   2060          call multip
81C0  114082   2070          ld   de,calc3
81C3  CD85BD   2080          call multip
81C6  CD2382   2090          call fin
81C9  C9       2100          ret
               2110 ;--* VOLUME D'UNE PYRAMIDE *--
81CA  3E07     2120 pyrami: ld   a,7
81CC  215982   2130          ld   hl,fonct
81CF  BE       2140          cp   (hl)
81D0  2021     2150          jr   nz,cylind
81D2  F1       2160          pop  af
81D3  FE03     2170          cp   3
81D5  C0       2180          ret  nz
81D6  AF       2190          xor  a
81D7  210300   2200          ld   hl,3
```

```
81DA  114082    2210          ld   de,calc3
81DD  CD64BD    2220          call entflo
81E0  213682    2230          ld   hl,calc1
81E3  113B82    2240          ld   de,calc2
81E6  CD85BD    2250          call multip
81E9  114082    2260          ld   de,calc3
81EC  CD88BD    2270          call divise
81EF  CD2382    2280          call fin
81F2  C9        2290          ret
                2300 ;--* VOLUME D'UN CYLINDRE *--
81F3  3E08      2310 cylind:  ld   a,8
81F5  215982    2320          ld   hl,fonct
81F8  BE        2330          cp   (hl)
81F9  C0        2340          ret  nz
81FA  F1        2350          pop  af
81FB  FE03      2360          cp   3
81FD  C0        2370          ret  nz
81FE  214082    2380          ld   hl,calc3
8201  CD9ABD    2390          call pi
8204  213682    2400          ld   hl,calc1
8207  113682    2410          ld   de,calc1
820A  CD85BD    2420          call multip
820D  113B82    2430          ld   de,calc2
8210  CD85BD    2440          call multip
8213  114082    2450          ld   de,calc3
8216  CD85BD    2460          call multip
8219  CD2382    2470          call fin
821C  C9        2480          ret
                2490 ;
821D  010500    2500 stock:   ld   bc,5
8220  EDB0      2510          ldir
8222  C9        2520          ret
8223  213682    2530 fin:     ld   hl,calc1
8226  ED5B3082  2540          ld   de,(sto1)
822A  010500    2550          ld   bc,5
822D  EDB0      2560          ldir
822F  C9        2570          ret
                2580 ;
8230            2590 sto1:    defs 2
8232            2600 sto2:    defs 2
8234            2610 sto3:    defs 2
8236            2620 calc1:   defs 5
823B            2630 calc2:   defs 5
8240            2640 calc3:   defs 5
8245            2650 calc4:   defs 5
824A            2660 calc5:   defs 5
824F            2670 calc6:   defs 5
8254            2680 calc7:   defs 5
8259            2690 fonct:   defs 2
```

```
Pass 2 errors: 00

al      8025  a2      803D  a3      805B
additi  BD7C  calc1   8236  calc2   823B
calc3   8240  calc4   8245  calc5   824A
calc6   824F  calc7   8254  conedt  8106
conetq  8140  copie   BD61  cube    8088
cylind  81F3  divise  BD88  entflo  BD64
ext     800A  fin     8223  fonct   8259
multip  BD85  nom     805D  parall  81AB
pi      BD9A  prisme  80ED  puiss   BDA0
pyrami  81CA  rout    8080  rsx     BCD1
space   8064  sphere  80A8  sto1    8230
sto2    8232  sto3    8234  stock   821D


Table used:   431  from   900
Executes: 32768
```

ÉLECTRICITÉ

NUMERO FONCTION	FIGURE	NOMBRE PARAM.	PARAMETRES	FORMULES
1	PUISSANCE ELECTRIQUE	2	R résist. I intensité	P = R×I^2
2	RESISTANCE	3	a résistiv. l longueur s section	$R = \frac{a \times l}{S}$
3	LOI D'OHM GENERATEUR	3	Re résis ext r résis int I intensité	E = (Re+r) × I
4	INDUCTION MAGNETIQUE	3	n nbr spire I intensité r ray spire	$B = \frac{2 \times pi \times 10^{-7} \times n \times I}{r}$
5	FLUX D'INDUCTION	3	S section B induction a angle	F = S×B×cos a
6	IMPEDANCE L/R	3	R résist. L inductan. w pulsation	$Z = \sqrt{(R^2+L^2 \times w^2)}$
7	IMPEDANCE C/R	3	R résist. C capacité w pulsation	$Z=\sqrt{(R^2+ \frac{1}{C^2 \times w^2})}$
8	RESISTANCE EQUIVALENTE	2	R1 résis 1 R2 résis 2	$R = \frac{R1 \times R2}{R1 + R2}$

Puissance électrique

|ELECTRICITE,1,@R,@I P = R×I^2

En entrée : R résistance.
I intensité.

En sortie : R puissance.

Cette formule est à rapprocher de la loi d'Ohm : U=RI, or P=UI donc P=RI×I=R×I^2.

Résistance

|ELECTRICITE,2,@a,@l,@s $R = a \times l/s$

En entrée : a résistivité.
l longueur.
s section de conducteur.

En sortie : a résistance.

La résistance d'un conducteur est proportionnelle à un coefficient appelé résistivité, liée à la matière formant le conducteur, et à sa longueur, mais elle est inversement proportionnelle à sa section. Cela signifie que plus la longueur est importante, plus la résistance est élevée, mais que plus la section augmente, plus la résistance diminue.

Loi d'Ohm pour un générateur

|ELECTRICITE,3,@Re,@r,@I $E = (Re+r) \times I$

En entrée : Re résistance du circuit.
r résistance interne du générateur.
I intensité.

En sortie : Re force électromotrice du générateur.

Pour un récepteur, la même loi se transforme pour devenir : $E = (Re-r) \times I$ où Re est la résistance du reste du circuit, alors que r est la résistance du récepteur.

Induction magnétique

|ELECTRICITE,4,@I,@r,@n $B = 2 \times pi \times 10\hat{}(-7) \times n \times I/r$

En entrée : I intensité.
r rayon d'une spire.
n nombre de tours de la bobine.

En sortie : I induction magnétique.

L'induction magnétique, mesurée en TESLA, est un champ magnétique créé par le passage d'un courant d'intensité I dans une bobine constituée de n spires dont le rayon fait r.

Flux d'induction

|ELECTRICITE,5,@a,@S,@B $F = S \times B \times \cos a$

En entrée : a angle.
S surface.
B induction.

En sortie : a flux d'induction.

Le flux d'induction magnétique est le produit scalaire entre le vecteur induction magnétique et le vecteur normal d'une surface. Il se mesure en WEBER.

Impédance d'un circuit bobine-résistance

|ELECTRICITE,6,@R,@L,@w $Z = \sqrt{(R^2 + L^2 \times w^2)}$

En entrée : R résistance.
L inductance.
w pulsation.

En sortie : R impédance.

L'impédance d'un circuit soumis à un courant sinusoïdal dépend des éléments qui le composent : résistance, bobine, capacité et pulsation (inverse de la période). Ce premier calcul concerne un circuit composé d'une résistance R et d'une bobine d'inductance L et soumis à un courant de pulsation w. Le suivant concerne un circuit dans lequel la bobine est remplacée par un condensateur de capacité C.

Impédance d'un circuit condensateur-résistance

|ELECTRICITE,7,@R,@C,@w $Z = \sqrt{(R^2 + 1/C^2 \times w^2)}$

En entrée : R résistance.
C capacité.
w pulsation.

En sortie : R impédance.

Résistance équivalente de 2 résistances en parallèle

|ELECTRICITE,8,@R1,@R2 $R = R1 \times R2/R1+R2$

En entrée : R1 première résistance.
R2 deuxième résistance.

En sortie : R1 résistance équivalente.

La formule générale de calcul de la résistance équivalente de n résistances en parralèle est : $1/R = 1/R1 + 1/R2 + ... + 1/Rn$. Ici elle est ramenée à deux résistances. Pour des résistances montées en série, la résistance équivalente est tout simplement obtenue en additionnant toutes les résistances.

Pour les condensateurs, les formules sont inversées. Si C est la capacité équivalente d'un réseau de condensateurs, on obtient pour un montage en série : $1/C = 1/C1 + 1/C2 + ... + 1/Cn$, et pour un montage en parallèle : $C = C1 + C2 + ... + Cn$.

La routine ci-dessus permet alors le calcul de la capacité Equivalente de deux condensateurs en série, en remplaçant R1 et R2 par C1 et C2.

Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                 10 ;=*  ROUTINE ELECTRICITE CPC 6128  *=
                 20 ;
8000             30         ent  #8000
8000             40         org  #8000
                 50 ;
                 60 ;--* TRAITEMENT RSX *--
                 70 ;
BCD1             80 rsx:    equ  #bcd1
8000  010A80     90         ld   bc,ext
8003  216980    100         ld   hl,space
8006  CDD1BC    110         call rsx
8009  C9        120         ret
800A  5D80      130 ext:    defw nom
                140 ;--* RECUPERATION DES PARAMETRES *--
                150 ;--* TEST SI 4 PARAMETRES *--
800C  F5        160         push af
800D  FE04      170         cp   4
800F  2014      180         jr   nz,a1
8011  DD6601    190         ld   h,(ix+1)
8014  DD6E00    200         ld   l,(ix+0)
8017  220E82    210         ld   (sto3),hl
801A  111A82    220         ld   de,calc3
801D  CDF781    230         call stock
8020  DD23      240         inc  ix
8022  DD23      250         inc  ix
8024  3D        260         dec  a
                270 ;--* TEST SI 3 PARAMETRES *--
8025  FE03      280 a1:     cp   3
8027  2014      290         jr   nz,a2
8029  DD6601    300         ld   h,(ix+1)
802C  DD6E00    310         ld   l,(ix+0)
802F  220C82    320         ld   (sto2),hl
8032  111582    330         ld   de,calc2
8035  CDF781    340         call stock
8038  DD23      350         inc  ix
803A  DD23      360         inc  ix
803C  3D        370         dec  a
                380 ;--* TEST SI 2 PARAMETRES *--
803D  FE02      390 a2:     cp   2
803F  201A      400         jr   nz,a3
8041  DD6601    410         ld   h,(ix+1)
8044  DD6E00    420         ld   l,(ix+0)
8047  220A82    430         ld   (sto1),hl
804A  111082    440         ld   de,calc1
804D  CDF781    450         call stock
8050  DD6603    460         ld   h,(ix+3)
```

```
8053  DD6E02     470          ld   l,(ix+2)
8056  222482     480          ld   (fonct),hl
8059  1825       490          jr   rout
805B  F1         500 a3:      pop  af
805C  C9         510          ret
                 520 ;
805D  454C4543   530 nom:     defm "ELECTRICIT"
8067  C5         540          defb "E"+#80
8068  00         550          defb 0
8069             560 space:   defs 4
                 570 ;
                 580 ;
                 590 ;--* TRAITEMENT CALL *--
                 600 ;
8080             610          org  #8080
                 620 ;
BD61             630 copie:   equ  #bd61
BD64             640 entflo:  equ  #bd64
BD79             650 puis10:  equ  #bd79
BD7C             660 additi:  equ  #bd7c
BD85             670 multip:  equ  #bd85
BD88             680 divise:  equ  #bd88
BD97             690 degres:  equ  #bd97
BD9A             700 pi:      equ  #bd9a
BD9D             710 racine:  equ  #bd9d
BDAF             720 cosin:   equ  #bdaf
                 730 ;
                 740 ;--* PUISSANCE *--
8080  3E01       750 rout:    ld   a,1
8082  212482     760          ld   hl,fonct
8085  BE         770          cp   (hl)
8086  2018       780          jr   nz,resist
8088  F1         790 puiss:   pop  af
8089  FE03       800          cp   3
808B  C0         810          ret  nz
808C  211582     820          ld   hl,calc2
808F  111582     830          ld   de,calc2
8092  CD85BD     840          call multip
8095  111082     850          ld   de,calc1
8098  EB         860          ex   de,hl
8099  CD85BD     870          call multip
809C  CDFD81     880          call fin
809F  C9         890          ret
                 900 ;--* RESISTANCE *--
80A0  3E02       910 resist:  ld   a,2
80A2  212482     920          ld   hl,fonct
80A5  BE         930          cp   (hl)
80A6  2017       940          jr   nz,ohm
80A8  F1         950          pop  af
```

```
80A9  FE04     960          cp   4
80AB  C0       970          ret  nz
80AC  211082   980          ld   hl,calc1
80AF  111582   990          ld   de,calc2
80B2  CD85BD   1000         call multip
80B5  111A82   1010         ld   de,calc3
80B8  CD88BD   1020         call divise
80BB  CDFD81   1030         call fin
80BE  C9       1040         ret
               1050 ;--* LOI D'OHM - GENERATEUR *--
80BF  3E03     1060 ohm:    ld   a,3
80C1  212482   1070         ld   hl,fonct
80C4  BE       1080         cp   (hl)
80C5  2017     1090         jr   nz,induct
80C7  F1       1100         pop  af
80C8  FE04     1110         cp   4
80CA  C0       1120         ret  nz
80CB  211082   1130         ld   hl,calc1
80CE  111582   1140         ld   de,calc2
80D1  CD7CBD   1150         call additi
80D4  111A82   1160         ld   de,calc3
80D7  CD85BD   1170         call multip
80DA  CDFD81   1180         call fin
80DD  C9       1190         ret
               1200 ;--* INDUCTION MAGNETIQUE *--
80DE  3E04     1210 induct: ld   a,4
80E0  212482   1220         ld   hl,fonct
80E3  BE       1230         cp   (hl)
80E4  2031     1240         jr   nz,flux
80E6  F1       1250         pop  af
80E7  FE04     1260         cp   4
80E9  C0       1270         ret  nz
80EA  211F82   1280         ld   hl,calc4
80ED  CD9ABD   1290         call pi
80F0  211082   1300         ld   hl,calc1
80F3  111582   1310         ld   de,calc2
80F6  CD88BD   1320         call divise
80F9  111A82   1330         ld   de,calc3
80FC  CD85BD   1340         call multip
80FF  111F82   1350         ld   de,calc4
8102  CD85BD   1360         call multip
8105  3EF9     1370         ld   a,#f9
8107  CD79BD   1380         call puis10
810A  211082   1390         ld   hl,calc1
810D  111082   1400         ld   de,calc1
8110  CD7CBD   1410         call additi
8113  CDFD81   1420         call fin
8116  C9       1430         ret
               1440 ;--* FLUX D'INDUCTION *--
```

```
8117  3E05      1450 flux:   ld   a,5
8119  212482    1460         ld   hl,fonct
811C  BE        1470         cp   (hl)
811D  201F      1480         jr   nz,impedl
811F  F1        1490         pop  af
8120  FE04      1500         cp   4
8122  C0        1510         ret  nz
8123  3E01      1520         ld   a,1
8125  CD97BD    1530         call degres
8128  211082    1540         ld   hl,calcl
812B  CDAFBD    1550         call cosin
812E  111582    1560         ld   de,calc2
8131  CD85BD    1570         call multip
8134  111A82    1580         ld   de,calc3
8137  CD85BD    1590         call multip
813A  CDFD81    1600         call fin
813D  C9        1610         ret
                1620 ;--* IMPEDANCE CIRCUIT L/R *--
813E  3E06      1630 impedl: ld   a,6
8140  212482    1640         ld   hl,fonct
8143  BE        1650         cp   (hl)
8144  2036      1660         jr   nz,imped2
8146  F1        1670         pop  af
8147  FE04      1680         cp   4
8149  C0        1690         ret  nz
814A  211082    1700         ld   hl,calcl
814D  111082    1710         ld   de,calcl
8150  CD85BD    1720         call multip
8153  211582    1730         ld   hl,calc2
8156  111582    1740         ld   de,calc2
8159  CD85BD    1750         call multip
815C  211A82    1760         ld   hl,calc3
815F  111A82    1770         ld   de,calc3
8162  CD85BD    1780         call multip
8165  211582    1790         ld   hl,calc2
8168  111A82    1800         ld   de,calc3
816B  CD85BD    1810         call multip
816E  111082    1820         ld   de,calcl
8171  EB        1830         ex   de,hl
8172  CD7CBD    1840         call additi
8175  CD9DBD    1850         call racine
8178  CDFD81    1860         call fin
817B  C9        1870         ret
                1880 ;--* IMPEDANCE CIRCUIT C/R *--
817C  3E07      1890 imped2: ld   a,7
817E  212482    1900         ld   hl,fonct
8181  BE        1910         cp   (hl)
8182  2043      1920         jr   nz,resieq
8184  F1        1930         pop  af
```

```
8185  FE04      1940          cp   4
8187  C0        1950          ret  nz
8188  AF        1960          xor  a
8189  210100    1970          ld   hl,1
818C  111F82    1980          ld   de,calc4
818F  CD64BD    1990          call entflo
8192  211582    2000          ld   hl,calc2
8195  111582    2010          ld   de,calc2
8198  CD85BD    2020          call multip
819B  211A82    2030          ld   hl,calc3
819E  111A82    2040          ld   de,calc3
81A1  CD85BD    2050          call multip
81A4  111582    2060          ld   de,calc2
81A7  CD85BD    2070          call multip
81AA  111F82    2080          ld   de,calc4
81AD  EB        2090          ex   de,hl
81AE  CD88BD    2100          call divise
81B1  211082    2110          ld   hl,calc1
81B4  111082    2120          ld   de,calc1
81B7  CD85BD    2130          call multip
81BA  111F82    2140          ld   de,calc4
81BD  CD7CBD    2150          call additi
81C0  CD9DBD    2160          call racine
81C3  CDFD81    2170          call fin
81C6  C9        2180          ret
                2190 ;--* RESISTANCE EQUIVALENTE *--
81C7  3E08      2200 resieq:  ld   a,8
81C9  212482    2210          ld   hl,fonct
81CC  BE        2220          cp   (hl)
81CD  C0        2230          ret  nz
81CE  F1        2240          pop  af
81CF  FE03      2250          cp   3
81D1  C0        2260          ret  nz
81D2  111082    2270          ld   de,calc1
81D5  211A82    2280          ld   hl,calc3
81D8  CD61BD    2290          call copie
81DB  211A82    2300          ld   hl,calc3
81DE  111582    2310          ld   de,calc2
81E1  CD7CBD    2320          call additi
81E4  211082    2330          ld   hl,calc1
81E7  111582    2340          ld   de,calc2
81EA  CD85BD    2350          call multip
81ED  111A82    2360          ld   de,calc3
81F0  CD88BD    2370          call divise
81F3  CDFD81    2380          call fin
81F6  C9        2390          ret
                2400 ;
81F7  010500    2410 stock:   ld   bc,5
81FA  EDB0      2420          ldir
```

```
81FC  C9        2430          ret
81FD  211082    2440 fin:     ld   hl,calc1
8200  ED5B0A82  2450          ld   de,(sto1)
8204  010500    2460          ld   bc,5
8207  EDB0      2470          ldir
8209  C9        2480          ret
                2490 ;
820A            2500 sto1:    defs 2
820C            2510 sto2:    defs 2
820E            2520 sto3:    defs 2
8210            2530 calc1:   defs 5
8215            2540 calc2:   defs 5
821A            2550 calc3:   defs 5
821F            2560 calc4:   defs 5
8224            2570 fonct:   defs 2

Pass 2 errors: 00

a1      8025  a2      803D  a3      805B
additi BD7C  calc1  8210  calc2  8215
calc3  821A  calc4  821F  copie  BD61
cosin  BDAF  degres BD97  divise BD88
entflo BD64  ext    800A  fin    81FD
flux   8117  fonct  8224  imped1 813E
imped2 817C  induct 80DE  multip BD85
nom    805D  ohm    80BF  pi     BD9A
puis10 BD79  puiss  8088  racine BD9D
resieq 81C7  resist 80A0  rout   8080
rsx    BCD1  space  8069  sto1   820A
sto2   820C  sto3   820E  stock  81F7


Table used:   430  from   900
Executes: 32768
```

PHYSIQUE

NUMERO FONCTION	FIGURE	NOMBRE PARAM	PARAMETRES	FORMULES
1	MOUVEMENT RECTILIGNE UNIFORME	4	a accélér. t temps Vo vit init Xo abs init	$X = \frac{a \times t^2}{2} + Vot + Xo$
2	MOUVEMENT RECTILIGNE SINUSOIDAL	3	a élong max w pulsation t temps	X = a sin (wt)
3	ENERGIE CINETIQUE	2	M masse V vitesse	$E : \frac{M \times V^2}{2}$
4	PERIODE D'OSCILLATION	1	l longueur	$T = 2pi \times \sqrt{(l/g)}$
5	PLAN INCLINE	2	P poids a angl incl	F = P sin a
6	DILATATION LINEAIRE	3	Lo long à 0 k coef dil. t température.	L = Lo (1+k×t)
7	TRAVAIL D'UNE FORCE	3	F intensité l longueur a angle	W = F×l× cos a
8	FORCE DE GRAVITATION	3	e=667×10^−13 M1 masse 1 M2 masse 2 d distance	$F = e \times \frac{M_1 \times M_2}{d^2}$

Equation d'un mouvement rectiligne uniforme

|PHYSIQUE,1,@t,@a,@Vo,@Xo $X = a \times t^2/2 + Vo \times t + Xo$

En entrée : t temps.
a accélération.
Vo vitesse initiale.
Xo abscisse à l'origine des temps.

En sortie : t abscisse au temps t.

Cette formule permet de trouver l'abscisse à un temps t d'un corps subissant une accélération constante a ayant à l'origine des temps une vitesse initiale Vo et une abscisse initiale Xo.

Une application simple de cette formule serait par exemple de calculer la profondeur d'un puits dans lequel on laisse tomber, sans vitesse initiale, une pierre qui met 2,5 secondes pour en atteindre le fond.

Il faut alors initialiser les variables comme suit :

t = 2.5 ; a = 9.81 (accélération de la pesanteur) ; Vo = 0 et Xo = 0. L'appel de la routine permettrait d'obtenir dans t la réponse demandée, à savoir 30,65 m.

Equation d'un mouvement rectiligne sinusoïdal

|PHYSIQUE,2,@a,@w,@t $X = a \times \sin(w \times t)$

En entrée : a élongation maximum.
w pulsation.
t temps.

En sortie : a abscisse au temps t.

Cette formule s'applique par exemple à un pendule ou à un objet lié à un ressort.

Energie cinétique

|PHYSIQUE,3,@V,@M $E = M \times V\hat{}2/2$

En entrée : V vitesse.
M masse.

En sortie : V énergie cinétique.

L'énergie cinétique représente l'énergie stockée par un corps mobile à une vitesse V. C'est aussi l'énergie qu'il faut fournir pour stopper le mouvement de ce corps. Elle est liée à la masse de celui-ci. Cela signifie qu'à vitesse égale, un corps plus lourd possédera une énergie supérieure à celle d'un corps plus léger.

Période d'un oscillateur

|PHYSIQUE,4,@a $T = 2 \times pi \times \sqrt{(l/g)}$

En entrée : l longueur.

En sortie : l période de l'oscillation.

g a pour valeur 10 m/s/s.

Cette formule permet de calculer la durée, séparant deux positions identiques, pour un pendule soumis à de faibles oscillations.

Plan incliné

|PHYSIQUE,5,@a,@P $F = P \times \sin a$

En entrée : a angle d'inclinaison de plan.
P poids du corps.

En sortie : a force parallèle au plan.

Très utilisée en statique, cette formule permet de déterminer l'intensité de la force à appliquer parallèlement au plan incliné, pour maintenir en équilibre un corps de poids P.

Par exemple, pour sortir une voiture d'une tonne (1000 kg donc à peu près 10000 N) tombée dans un fossé de pente 30 degrés, il faut exercer sur elle une force de traction, parallèle au plan incliné, d'intensité 5000 N (la moitié du poids de la voiture). Ce résultat est obtenu après utilisation de la routine dont les variables déclarées sont : a = 30 et P = 10000.

Bien entendu, nous n'avons pas tenu compte des frottements qui, dans un cas concret, ne manqueraient pas d'intervenir.

Dilatation linéaire

|PHYSIQUE,6,@Lo,@k,@t $L = Lo(1+k \times t)$

En entrée : Lo longueur à 0 degré.
k coefficient de dilatation linéaire.
t température.

En sortie : Lo longueur à c degrés.

Tout corps subit une dilatation lors d'une augmentation de température. Cette dilatation dépend d'un coefficient propre au matériau, de la longueur à 0 degré de ce corps et de l'augmentation de la température.

De même, nous pourrions calculer les dilatations volumiques d'un corps.

Travail d'une force

|PHYSIQUE,7,@F,@l,@a $W = F \times l \times \cos a$

En entrée : F intensité de la force.
l longueur de déplacement du point d'application.
a angle force/déplacement.

En sortie : F travail de la force.

Force de gravitation

|PHYSIQUE,8,@M1,@M2,@d $F = 6.67 \times 10\hat{}(-11) \times M_1 \times M_2/d\hat{}2$

En entrée : M1 masse du corps 1.
M2 masse du corps 2.
d distance entre les 2 corps.

En sortie : d intensité de la force de la gravitation.

C'est une pomme dit-on qui inspira à NEWTON, lors d'une sieste, la formulation de la loi de l'attraction universelle représentée par cette égalité. Elle lui permit également de retrouver les lois de Kepler relatives au système planétaire. Ces lois précisent que les planètes s'attirent entre elles avec une force proportionnelle à leurs masses et inversement proportionnelle au carré de leur distance.

Noter que dans la routine le coefficient devient 667 × 10 puissance −13.

```
Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

                  10 ;=*  ROUTINE PHYSIQUE CPC 464  *=
                  20 ;
8000              30          ent  #8000
8000              40          org  #8000
                  50 ;
                  60 ;--* TRAITEMENT RSX *--
                  70 ;
BCD1              80 rsx:     equ  #bcd1
8000  010A80      90          ld   bc,ext
8003  217E80     100          ld   hl,space
8006  CDD1BC     110          call rsx
8009  C9         120          ret
800A  7580       130 ext:     defw nom
                 140 ;--* RECUPERATION DES PARAMETRES *--
                 150 ;--* TEST SI 5 PARAMETRES *--
800C  F5         160          push af
800D  FE05       170          cp   5
800F  2014       180          jr   nz,a0
8011  DD6601     190          ld   h,(ix+1)
8014  DD6E00     200          ld   l,(ix+0)
8017  221E82     210          ld   (sto4),hl
801A  112F82     220          ld   de,calc4
801D  CD0582     230          call stock
8020  DD23       240          inc  ix
8022  DD23       250          inc  ix
8024  3D         260          dec  a
                 270 ;--* TEST SI 4 PARAMETRES *--
8025  FE04       280 a0:      cp   4
8027  2014       290          jr   nz,a1
8029  DD6601     300          ld   h,(ix+1)
802C  DD6E00     310          ld   l,(ix+0)
802F  221C82     320          ld   (sto3),hl
8032  112A82     330          ld   de,calc3
8035  CD0582     340          call stock
8038  DD23       350          inc  ix
803A  DD23       360          inc  ix
```

```
803C  3D        370          dec  a
                380 ;--* TEST SI 3 PARAMETRES *--
803D  FE03      390 a1:      cp   3
803F  2014      400          jr   nz,a2
8041  DD6601    410          ld   h,(ix+1)
8044  DD6E00    420          ld   l,(ix+0)
8047  221A82    430          ld   (sto2),hl
804A  112582    440          ld   de,calc2
804D  CD0582    450          call stock
8050  DD23      460          inc  ix
8052  DD23      470          inc  ix
8054  3D        480          dec  a
                490 ;--* TEST SI 2 PARAMETRES *--
8055  FE02      500 a2:      cp   2
8057  201A      510          jr   nz,a3
8059  DD6601    520          ld   h,(ix+1)
805C  DD6E00    530          ld   l,(ix+0)
805F  221882    540          ld   (sto1),hl
8062  112082    550          ld   de,calc1
8065  CD0582    560          call stock
8068  DD6603    570          ld   h,(ix+3)
806B  DD6E02    580          ld   l,(ix+2)
806E  223982    590          ld   (fonct),hl
8071  181D      600          jr   rout
8073  F1        610 a3:      pop  af
8074  C9        620          ret
                630 ;
8075  50485953  640 nom:     defm "PHYSIQU"
807C  C5        650          defb "E"+#80
807D  00        660          defb 0
807E            670 space:   defs 4
                680 ;
                690 ;
                700 ;--* TRAITEMENT CALL *--
                710 ;
8090            720          org  #8090
                730 ;
BD64            740 entflo:  equ  #bd64
BD79            750 puis10:  equ  #bd79
BD7C            760 additi:  equ  #bd7c
BD85            770 multip:  equ  #bd85
BD88            780 divise:  equ  #bd88
BD97            790 degres:  equ  #bd97
BD9A            800 pi:      equ  #bd9a
BD9D            810 racine:  equ  #bd9d
BDAC            820 sinus:   equ  #bdac
BDAF            830 cosin:   equ  #bdaf
                840 ;
8090  AF        850 rout:    xor  a
```

```
8091  210200    860          ld   hl,2
8094  113482    870          ld   de,deux
8097  CD64BD    880          call entflo
                890 ;
                900 ;--* MOUVEMENT RECTILIGNE *--
809A  3E01      910 mvtrec: ld   a,1
809C  213982    920          ld   hl,fonct
809F  BE        930          cp   (hl)
80A0  2032      940          jr   nz,mvtsin
80A2  F1        950          pop  af
80A3  FE05      960          cp   5
80A5  C0        970          ret  nz
80A6  212A82    980          ld   hl,calc3
80A9  112082    990          ld   de,calc1
80AC  CD85BD   1000          call multip
80AF  212082   1010          ld   hl,calc1
80B2  112082   1020          ld   de,calc1
80B5  CD85BD   1030          call multip
80B8  112582   1040          ld   de,calc2
80BB  CD85BD   1050          call multip
80BE  113482   1060          ld   de,deux
80C1  CD88BD   1070          call divise
80C4  112A82   1080          ld   de,calc3
80C7  CD7CBD   1090          call additi
80CA  112F82   1100          ld   de,calc4
80CD  CD7CBD   1110          call additi
80D0  CD0B82   1120          call fin
80D3  C9       1130          ret
               1140 ;--* MOUVEMENT RECTILIGNE SINUSOIDAL *--
80D4  3E02     1150 mvtsin: ld   a,2
80D6  213982   1160          ld   hl,fonct
80D9  BE       1170          cp   (hl)
80DA  2020     1180          jr   nz,enerci
80DC  F1       1190          pop  af
80DD  FE04     1200          cp   4
80DF  C0       1210          ret  nz
80E0  3E01     1220          ld   a,1
80E2  CD97BD   1230          call degres
80E5  212582   1240          ld   hl,calc2
80E8  112A82   1250          ld   de,calc3
80EB  CD85BD   1260          call multip
80EE  CDACBD   1270          call sinus
80F1  112082   1280          ld   de,calc1
80F4  EB       1290          ex   de,hl
80F5  CD85BD   1300          call multip
80F8  CD0B82   1310          call fin
80FB  C9       1320          ret
               1330 ;--* ENERGIE CINETIQUE *--
80FC  3E03     1340 enerci: ld   a,3
```

```
80FE  213982   1350          ld   hl,fonct
8101  BE       1360          cp   (hl)
8102  201E     1370          jr   nz,perosc
8104  F1       1380          pop  af
8105  FE03     1390          cp   3
8107  C0       1400          ret  nz
8108  212582   1410          ld   hl,calc2
810B  112582   1420          ld   de,calc2
810E  CD85BD   1430          call multip
8111  112082   1440          ld   de,calc1
8114  EB       1450          ex   de,hl
8115  CD85BD   1460          call multip
8118  113482   1470          ld   de,deux
811B  CD88BD   1480          call divise
811E  CD0B82   1490          call fin
8121  C9       1500          ret
               1510 ;--* PERIODE D'OSCILLATEUR *--
8122  3E04     1520 perosc: ld   a,4
8124  213982   1530          ld   hl,fonct
8127  BE       1540          cp   (hl)
8128  2030     1550          jr   nz,plainc
812A  F1       1560          pop  af
812B  FE02     1570          cp   2
812D  C0       1580          ret  nz
812E  212582   1590          ld   hl,calc2
8131  CD9ABD   1600          call pi
8134  AF       1610          xor  a
8135  210A00   1620          ld   hl,10
8138  112A82   1630          ld   de,calc3
813B  CD64BD   1640          call entflo
813E  212082   1650          ld   hl,calc1
8141  112A82   1660          ld   de,calc3
8144  CD88BD   1670          call divise
8147  CD9DBD   1680          call racine
814A  112582   1690          ld   de,calc2
814D  CD85BD   1700          call multip
8150  113482   1710          ld   de,deux
8153  CD85BD   1720          call multip
8156  CD0B82   1730          call fin
8159  C9       1740          ret
               1750 ;--* PLAN INCLINE *--
815A  3E05     1760 plainc: ld   a,5
815C  213982   1770          ld   hl,fonct
815F  BE       1780          cp   (hl)
8160  2019     1790          jr   nz,dilat
8162  F1       1800          pop  af
8163  FE03     1810          cp   3
8165  C0       1820          ret  nz
8166  3E01     1830          ld   a,1
```

```
8168  CD97BD   1840          call degres
816B  212082   1850          ld   hl,calc1
816E  CDACBD   1860          call sinus
8171  112582   1870          ld   de,calc2
8174  CD85BD   1880          call multip
8177  CD0B82   1890          call fin
817A  C9       1900          ret
               1910 ;--* DILATION LINEAIRE *--
817B  3E06     1920 dilat:   ld   a,6
817D  213982   1930          ld   hl,fonct
8180  BE       1940          cp   (hl)
8181  201E     1950          jr   nz,travai
8183  F1       1960          pop  af
8184  FE04     1970          cp   4
8186  C0       1980          ret  nz
8187  212582   1990          ld   hl,calc2
818A  112082   2000          ld   de,calc1
818D  CD85BD   2010          call multip
8190  112A82   2020          ld   de,calc3
8193  CD85BD   2030          call multip
8196  112082   2040          ld   de,calc1
8199  EB       2050          ex   de,hl
819A  CD7CBD   2060          call additi
819D  CD0B82   2070          call fin
81A0  C9       2080          ret
               2090 ;--* TRAVAIL D'UNE FORCE *--
81A1  3E07     2100 travai:  ld   a,7
81A3  213982   2110          ld   hl,fonct
81A6  BE       2120          cp   (hl)
81A7  2020     2130          jr   nz,gravit
81A9  F1       2140          pop  af
81AA  FE04     2150          cp   4
81AC  3E01     2160          ld   a,1
81AE  CD97BD   2170          call degres
81B1  C0       2180          ret  nz
81B2  212A82   2190          ld   hl,calc3
81B5  CDAFBD   2200          call cosin
81B8  112582   2210          ld   de,calc2
81BB  CD85BD   2220          call multip
81BE  112082   2230          ld   de,calc1
81C1  EB       2240          ex   de,hl
81C2  CD85BD   2250          call multip
81C5  CD0B82   2260          call fin
81C8  C9       2270          ret
               2280 ;--* FORCE DE GRAVITATION *--
81C9  3E08     2290 gravit:  ld   a,8
81CB  213982   2300          ld   hl,fonct
81CE  BE       2310          cp   (hl)
81CF  C0       2320          ret  nz
```

```
81D0  F1        2330          pop  af
81D1  FE04      2340          cp   4
81D3  C0        2350          ret  nz
81D4  AF        2360          xor  a
81D5  219B02    2370          ld   hl,667
81D8  112F82    2380          ld   de,calc4
81DB  CD64BD    2390          call entflo
81DE  212082    2400          ld   hl,calc1
81E1  112582    2410          ld   de,calc2
81E4  CD85BD    2420          call multip
81E7  112A82    2430          ld   de,calc3
81EA  CD88BD    2440          call divise
81ED  112A82    2450          ld   de,calc3
81F0  CD88BD    2460          call divise
81F3  112F82    2470          ld   de,calc4
81F6  CD85BD    2480          call multip
81F9  3EF3      2490          ld   a,#f3
81FB  212082    2500          ld   hl,calc1
81FE  CD79BD    2510          call puis10
8201  CD0B82    2520          call fin
8204  C9        2530          ret
                2540 ;
8205  010500    2550 stock:   ld   bc,5
8208  EDB0      2560          ldir
820A  C9        2570          ret
820B  212082    2580 fin:     ld   hl,calc1
820E  ED5B1882  2590          ld   de,(sto1)
8212  010500    2600          ld   bc,5
8215  EDB0      2610          ldir
8217  C9        2620          ret
                2630 ;
8218            2640 sto1:    defs 2
821A            2650 sto2:    defs 2
821C            2660 sto3:    defs 2
821E            2670 sto4:    defs 2
8220            2680 calc1:   defs 5
8225            2690 calc2:   defs 5
822A            2700 calc3:   defs 5
822F            2710 calc4:   defs 5
8234            2720 deux:    defs 5
8239            2730 fonct:   defs 2
```

```
Pass 2 errors: 00

a0      8025  a1      803D  a2      8055
a3      8073  additi  BD7C  calc1   8220
calc2   8225  calc3   822A  calc4   822F
cosin   BDAF  degres  BD97  deux    8234
dilat   817B  divise  BD88  enerci  80FC
entflo  BD64  ext     800A  fin     820B
fonct   8239  gravit  81C9  multip  BD85
mvtrec  809A  mvtsin  80D4  nom     8075
perosc  8122  pi      BD9A  plainc  815A
puis10  BD79  racine  BD9D  rout    8090
rsx     BCD1  sinus   BDAC  space   807E
sto1    8218  sto2    821A  sto3    821C
sto4    821E  stock   8205  travai  81A1


Table used:   466  from   900
Executes: 32768
```

MATHEMATIQUES

NUMERO FONCTION	OBJET	NOMBRE PARAM	PARAMETRES	FORMULES
1	EQUATION DU SECOND DEGRE	3	a coef x^2 b coef x^1 c coef x^0	$d = \sqrt{(b^2-4ac)}$ $x1=(-b-d/2a)$ $x2=(-b+d/2a)$
2	FACTORIEL	1	a nbre ent	$f=a\times a-1\times\ldots\times 1$
3	SOMME SUITE ARITHMETIQUE	3	n nbr terme U1 1er term Un der term	$S=n(U1+Un)/2$
4	SOMME SUITE GEOMETRIQUE	3	n nbr terme r raison a1 1er term	$S = \frac{U1\ (1-r^n)}{(1-r)}$

Résolution d'équations du second degré

|MATH,1,@a,@b,@c $\quad x1 = -b-\sqrt{(d)/2\times a}$ avec $d=b^2-4ac$

$x2 = -b+\sqrt{(d)/2\times a}$

En entrée : a coefficient de x^2.
b coefficient de x^1.
c coefficient de x^0.

En sortie : a valeur de d si celui-ci est négatif.
b première solution.
c deuxième solution.

Une équation du second degré se présente sous la forme :

$$a\times x^2 + b\times x + c = 0$$

Pour obtenir les solutions de l'équation, il faut calculer le disciminant $d = b^2 - 4\times a\times c$. Pour d supérieur à 0, il existe deux solutions réelles x1 et x2 dont les valeurs sont données par les formules ci-dessus. Pour $d = 0$, $x1 = x2 = -b/2\times a$ et pour d plus petit que 0, il n'existe pas de solutions réelles.

La routine débute par le transfert des 3 coefficients dans CALC4, CALC5 et CALC6 car ils seront appelés à servir plusieurs fois. Le premier calcul consiste à chercher le discriminant. Il faut ensuite tester son signe. S'il est négatif, la routine place sa valeur dans a et s'arrête là. S'il est positif, les deux solutions sont calculées et placées dans b et c.

Factoriel

|MATH,2,@a $f = a \times a-1 \times a-2 \times a-3 \times ... \times 1$

En entrée : a nombre dont on cherche le factoriel.

En sortie : a factoriel du nombre.

Les factoriels sont des fonctions utilisées notamment dans les calculs statistiques, par exemple dans le cas des combinaisons et des arrangements. Ce sont ces calculs qui permettent de trouver le nombre de combinaisons qu'il faudrait jouer au tiercé pour être sûr de gagner.

Somme des éléments d'une suite arithmétique

|MATH,3,@U1,@Un,@n $S = n \times (U1+Un)/2$

En entrée : a premier terme de la suite.
b dernier terme de la suite.
c nombre de termes de la suite.

En sortie : a somme des termes.

Une suite arithmétique est formée de nombres caractérisés de la manière suivante : $U(i) = U(i-1) + r$ où r s'appelle la raison. La somme de ces nombres s'obtient par la formule ci-dessus.

La connaissance de cette formule permet d'effectuer des calculs rapides. Par exemple, la somme des nombres de 1 à 1000 se trouve très vite : $S = 1000 \times (1000+1)/2 = 500500$

Somme des éléments d'une suite géométrique

|MATH,4,@U1,@r,@n $S = U1 \times (1-r\hat{}n)/(1-r)$

En entrée : U1 premier terme de la suite.
r raison de la suite.
n nombre de termes de la suite.

En sortie : a somme des termes.

La caractéristique d'un terme d'une suite géométrique est d'être calculée par multiplication du terme précédent par un nombre appelé raison de la suite : $U(i) = U(i-1) \times r$. La somme d'une suite de ce type comprenant n nombres est obtenue par la formule décrite plus haut.

Hisoft GENA3.1 Assembler.

Pass 1 errors: 00

```
                 10 ;=*  ROUTINE MATH CPC 6128  *=
                 20 ;
8000             30          ent  #8000
8000             40          org  #8000
                 50 ;
                 60 ;--* TRAITEMENT RSX *--
                 70 ;
BCD1             80 rsx:     equ  #bcd1
8000  010A80     90          ld   bc,ext
8003  216280    100          ld   hl,space
8006  CDD1BC    110          call rsx
8009  C9        120          ret
800A  5D80      130 ext:     defw nom
                140 ;--* RECUPERATION DES PARAMETRES *--
                150 ;--* TEST SI 4 PARAMETRES *--
800C  F5        160          push af
800D  FE04      170          cp   4
800F  2014      180          jr   nz,a1
8011  DD6601    190          ld   h,(ix+1)
8014  DD6E00    200          ld   l,(ix+0)
8017  221082    210          ld   (sto3),hl
801A  111C82    220          ld   de,calc3
801D  CDF981    230          call stock
8020  DD23      240          inc  ix
8022  DD23      250          inc  ix
8024  3D        260          dec  a
                270 ;--* TEST SI 3 PARAMETRES *--
8025  FE03      280 a1:      cp   3
8027  2014      290          jr   nz,a2
8029  DD6601    300          ld   h,(ix+1)
802C  DD6E00    310          ld   l,(ix+0)
802F  220E82    320          ld   (sto2),hl
8032  111782    330          ld   de,calc2
8035  CDF981    340          call stock
8038  DD23      350          inc  ix
803A  DD23      360          inc  ix
803C  3D        370          dec  a
                380 ;--* TEST SI 2 PARAMETRES *--
803D  FE02      390 a2:      cp   2
803F  201A      400          jr   nz,a3
8041  DD6601    410          ld   h,(ix+1)
8044  DD6E00    420          ld   l,(ix+0)
8047  220C82    430          ld   (sto1),hl
804A  111282    440          ld   de,calc1
804D  CDF981    450          call stock
```

```
8050  DD6603    460          ld   h,(ix+3)
8053  DD6E02    470          ld   l,(ix+2)
8056  223582    480          ld   (fonct),hl
8059  1825      490          jr   rout
805B  F1        500 a3:      pop  af
805C  C9        510          ret
                520 ;
805D  4D4154    530 nom:     defm "MAT"
8060  C8        540          defb "H"+#80
8061  00        550          defb 0
8062            560 space:   defs 4
                570 ;
                580 ;
                590 ;--* TRAITEMENT CALL *--
                600 ;
8080            610          org  #8080
                620 ;
BD61            630 copie:   equ  #bd61
BD64            640 entflo:  equ  #bd64
BD7C            650 additi:  equ  #bd7c
BD82            660 soustr:  equ  #bd82
BD85            670 multip:  equ  #bd85
BD88            680 divise:  equ  #bd88
BD8E            690 compar:  equ  #bd8e
BD91            700 invsig:  equ  #bd91
BD94            710 testsi:  equ  #bd94
BD9D            720 racine:  equ  #bd9d
BDA0            730 puiss:   equ  #bda0
                740 ;
                750 ;--* EQUATION DU SECOND DEGRE *--
                760 ;--* CALCUL DU DISCRIMINANT *--
8080  3E01      770 rout:    ld   a,1
8082  213582    780          ld   hl,fonct
8085  BE        790          cp   (hl)
8086  205F      800          jr   nz,facto
8088  F1        810          pop  af
8089  FE04      820          cp   4
808B  C0        830          ret  nz
808C  AF        840          xor  a
808D  210400    850          ld   hl,4
8090  113082    860          ld   de,calc7
8093  CD64BD    870          call entflo
8096  111282    880          ld   de,calc1
8099  212182    890          ld   hl,calc4
809C  CD61BD    900          call copie
809F  111782    910          ld   de,calc2
80A2  212682    920          ld   hl,calc5
80A5  CD61BD    930          call copie
80A8  111C82    940          ld   de,calc3
```

```
80AB  212B82     950          ld   hl,calc6
80AE  CD61BD     960          call copie
80B1  212682     970          ld   hl,calc5
80B4  111782     980          ld   de,calc2
80B7  CD85BD     990          call multip
80BA  212182    1000          ld   hl,calc4
80BD  112B82    1010          ld   de,calc6
80C0  CD85BD    1020          call multip
80C3  113082    1030          ld   de,calc7
80C6  EB        1040          ex   de,hl
80C7  CD85BD    1050          call multip
80CA  CD91BD    1060          call invsig
80CD  112682    1070          ld   de,calc5
80D0  CD7CBD    1080          call additi
                1090 ;--* TEST DU SIGNE DU DISCRIMINANT *--
80D3  CD94BD    1100          call testsi
80D6  FEFF      1110          cp   #ff
80D8  200F      1120          jr   nz,a4
80DA  113082    1130          ld   de,calc7
80DD  211282    1140          ld   hl,calc1
80E0  CD61BD    1150          call copie
80E3  CDFF81    1160          call fin
80E6  C9        1170          ret
                1180 ;--* CALCUL DES SOLUTIONS *--
80E7  1852      1190 facto:   jr   factor
80E9  213082    1200 a4:      ld   hl,calc7
80EC  CD9DBD    1210          call racine
80EF  111782    1220          ld   de,calc2
80F2  212682    1230          ld   hl,calc5
80F5  CD61BD    1240          call copie
80F8  211282    1250          ld   hl,calc1
80FB  111282    1260          ld   de,calc1
80FE  CD7CBD    1270          call additi
                1280 ;--* CALCUL SOLUTION 1 *--
8101  211782    1290          ld   hl,calc2
8104  113082    1300          ld   de,calc7
8107  CD7CBD    1310          call additi
810A  111282    1320          ld   de,calc1
810D  CD88BD    1330          call divise
8110  CD91BD    1340          call invsig
8113  211782    1350          ld   hl,calc2
8116  ED5B0E82  1360          ld   de,(sto2)
811A  010500    1370          ld   bc,5
811D  EDB0      1380          ldir
811F  212682    1390          ld   hl,calc5
8122  CD91BD    1400          call invsig
                1410 ;--* CALCUL SOLUTION 2 *--
8125  113082    1420          ld   de,calc7
8128  CD7CBD    1430          call additi
```

```
812B  111282    1440          ld   de,calc1
812E  CD88BD    1450          call divise
8131  ED5B1082  1460          ld   de,(sto3)
8135  010500    1470          ld   bc,5
8138  EDB0      1480          ldir
813A  C9        1490          ret
                1500 ;--* FACTORIEL *--
813B  3E02      1510 factor: ld   a,2
813D  213582    1520          ld   hl,fonct
8140  BE        1530          cp   (hl)
8141  203E      1540          jr   nz,suite1
8143  F1        1550          pop  af
8144  FE02      1560          cp   2
8146  C0        1570          ret  nz
8147  AF        1580          xor  a
8148  210100    1590          ld   hl,1
814B  113782    1600          ld   de,un
814E  CD64BD    1610          call entflo
8151  111282    1620          ld   de,calc1
8154  211782    1630          ld   hl,calc2
8157  CD61BD    1640          call copie
815A  211782    1650 a5:      ld   hl,calc2
815D  113782    1660          ld   de,un
8160  CD82BD    1670          call soustr
8163  DD211782  1680          ld   ix,calc2
8167  DDCB03BE  1690          res  7,(ix+3)
816B  DD7E04    1700          ld   a,(ix+4)
816E  FE81      1710          cp   #81
8170  280B      1720          jr   z,a6
8172  211282    1730          ld   hl,calc1
8175  111782    1740          ld   de,calc2
8178  CD85BD    1750          call multip
817B  18DD      1760          jr   a5
817D  CDFF81    1770 a6:      call fin
8180  C9        1780          ret
                1790 ;--* SOMME D'UNE SUITE ARITHMETIQUE *--
8181  3E03      1800 suite1: ld   a,3
8183  213582    1810          ld   hl,fonct
8186  BE        1820          cp   (hl)
8187  2027      1830          jr   nz,suite2
8189  F1        1840          pop  af
818A  FE04      1850          cp   4
818C  C0        1860          ret  nz
818D  AF        1870          xor  a
818E  210200    1880          ld   hl,2
8191  112182    1890          ld   de,calc4
8194  CD64BD    1900          call entflo
8197  211282    1910          ld   hl,calc1
819A  111782    1920          ld   de,calc2
```

```
819D  CD7CBD    1930          call additi
81A0  111C82    1940          ld   de,calc3
81A3  CD85BD    1950          call multip
81A6  112182    1960          ld   de,calc4
81A9  CD88BD    1970          call divise
81AC  CDFF81    1980          call fin
81AF  C9        1990          ret
                2000 ;--* SOMME D'UNE SUITE GEOMETRIQUE *--
81B0  3E04      2010 suite2:  ld   a,4
81B2  213582    2020          ld   hl,fonct
81B5  BE        2030          cp   (hl)
81B6  C0        2040          ret  nz
81B7  F1        2050          pop  af
81B8  FE04      2060          cp   4
81BA  C0        2070          ret  nz
81BB  AF        2080          xor  a
81BC  210100    2090          ld   hl,1
81BF  112182    2100          ld   de,calc4
81C2  CD64BD    2110          call entflo
81C5  111782    2120          ld   de,calc2
81C8  212682    2130          ld   hl,calc5
81CB  CD61BD    2140          call copie
81CE  211782    2150          ld   hl,calc2
81D1  111C82    2160          ld   de,calc3
81D4  CDA0BD    2170          call puiss
81D7  112182    2180          ld   de,calc4
81DA  CD82BD    2190          call soustr
81DD  212682    2200          ld   hl,calc5
81E0  112182    2210          ld   de,calc4
81E3  CD82BD    2220          call soustr
81E6  211282    2230          ld   hl,calc1
81E9  111782    2240          ld   de,calc2
81EC  CD85BD    2250          call multip
81EF  112682    2260          ld   de,calc5
81F2  CD88BD    2270          call divise
81F5  CDFF81    2280          call fin
81F8  C9        2290          ret
                2300 ;
81F9  010500    2310 stock:   ld   bc,5
81FC  EDB0      2320          ldir
81FE  C9        2330          ret
81FF  211282    2340 fin:     ld   hl,calc1
8202  ED5B0C82  2350          ld   de,(sto1)
8206  010500    2360          ld   bc,5
8209  EDB0      2370          ldir
820B  C9        2380          ret
                2390 ;
820C            2400 sto1:    defs 2
820E            2410 sto2:    defs 2
```

```
8210            2420 sto3:   defs 2
8212            2430 calc1:  defs 5
8217            2440 calc2:  defs 5
821C            2450 calc3:  defs 5
8221            2460 calc4:  defs 5
8226            2470 calc5:  defs 5
822B            2480 calc6:  defs 5
8230            2490 calc7:  defs 5
8235            2500 fonct:  defs 2
8237            2510 un:     defs 5


Pass 2 errors: 00

a1      8025  a2      803D  a3      805B
a4      80E9  a5      815A  a6      817D
additi BD7C  calc1   8212  calc2   8217
calc3   821C  calc4   8221  calc5   8226
calc6   822B  calc7   8230  compar BD8E
copie   BD61  divise BD88  entflo BD64
ext     800A  facto   80E7  factor 813B
fin     81FF  fonct   8235  invsig BD91
multip BD85  nom     805D  puiss  BDA0
racine BD9D  rout    8080  rsx    BCD1
soustr BD82  space   8062  sto1   820C
sto2    820E  sto3    8210  stock  81F9
suite1 8181  suite2 81B0  testsi BD94
un      8237

Table used:   472  from   900
Executes: 32768
```

ANNEXE 1

UTILISATION, IMPLANTATION, LONGUEUR DES ROUTINES

NOM ET FONCTION DES ROUTINES NON RSX

Les seules routines NON RSX se trouvent dans le chapitre 2 sur le traitement des nombres.

NOM ROUTINE	FONCTION
AFFBIN	affichage binaire
AFFHEX	affichage hexa décimal
AFFNUM	affichage entier non signé (avec vecteurs)
AFFENTS	affichage entier signé (avec vecteurs)
ASCII	affichage entier non signé
ASCIIS	affichage entier signé
AFFLOT	affichage flottant (avec vecteurs)
ENTFLO	récupération partie entière d'un flottant
DECFLO	récupération partie décimale d'un flottant

ROUTINES DE TRAITEMENT DU FLOTTANT	MANTISSE				EXPOS.
	oct 1	oct 2	oct 3	oct 4	oct 5
AFFLOT / ENTFLO / DECFLO	# 7008	# 700D	# 7012	# 7017	# 701C

UTILISATION DES ROUTINES NON RSX

NOM ROUTINE	ENTREE		ADRESSE BUFFER en SORTIE	ADRESSE pour CALL
	VALEUR	IMPLANTATION		
AFFBIN	entière	#7034 et #7035	#7036 à #7045	#7000
AFFHEX	entière	#7044 et #7045	#7046 à #7049	#7000
AFFNUM	entière	#70ED et #70EE	#70EF à #70F3	#7000
AFFENTS	entière	#710F et #7110	#7111 à #7116	#7000
ASCII	entière	#7043 et #7044	#7045 à #704A	#7000
ASCIIS	entière	#706A et #706B	#706C à #7072	#7000
AFFLOT	flottante	#7213 à #7217	#721C à #7221	#7020
ENTFLO	flottante	#709A à #709E	#709F à #70A0	#7020
DECFLO	flottante	#70EA à #70EE	#70EF à #70F0	#7020

NOM ET FONCTION DES ROUTINES RSX

NOM ROUTINE	FONCTION
FIND	recherche d'une chaîne de caractères
CHANGE	remplacement d'une chaîne de caractères
TRIC	tri de chaînes de caractères
TRIE	tri de nombres entiers
TRIR	tri de nombres réels
COULGRAF	encres du stylo et du papier graphiques
FENGRAF	création d'une fenêtre pour les graphiques
TRIANGLE	tracé de triangle quelconque
RECTANGLE 1	tracé de rectangle
RECTANGLE 2	tracé de rectangle (avec orientation)
CERCLE	tracé de cercle
ELLIPSE	tracé d'ellipse (avec orientation)
POLYGONE	tracé de polygone régulier (avec orientation)
COPZONE	copie d'une zone d'écran
ZOOM	agrandissement d'une zone de l'écran
SUPZOOM	agrandissement d'une zone avec déplacement
AIRE	calculs d'aire
VOLUME	calculs de volume
ELECTRICITE	calculs d'électricité
PHYSIQUE	calculs de physique
MATH	calculs de maths

APPEL RSX AVEC PASSAGE DE PARAMETRES

Rappel

L'appel d'une fonction RSX se fait de la manière suivante :

|NOM RSX,parm 1, ... ,parm N

NOM RSX	PARAMETRES (dans l'ordre indiqué)
FIND	adr déb zone, adr fin zone, adr txt voulu
CHANGE	adr déb zone, adr fin zone, adr txt anc, adr txt nouv
TRIC	adr 1er élément, [nb élément à trier]
TRIE	adr 1er élément, [nb élément à trier]
TRIR	adr 1er élément, [nb élément à trier]
COULGRAF	encore stylo, [encre papier]
FENGRAF	X bord gauche, X bord droit, Y bord haut, Y bord bas
TRIANGLE	X som 1, Y som 1, X som 2, Y som 2, X som 3, Y som 3
RECTANGLE (1)	X pt dépt, Y pt dépt, lg 1er cote, 1 g 2e cote
RECTANGLE (2)	X pt dépt, Y pt dépt, lg 1er cote, lg 2e cote, angle
CERCLE	X centre, Y centre, rayon
ELLIPSE	X centre, Y centre, lg 1er mi-axe, lg 2e mi-axe, angle
POLYGONE	nb cotes, X centre, Y centre, rayon, angle au 1er sommet
COPZONE	X départ, Y départ, lg zone, haut zone, X arrivé, Y arrivé
ZOOM	X départ (bas gauche), Y départ (bas gauche)
SUPZOOM	X départ (bas gauche), Y départ (bas gauche)
AIRE	numéro fonction, paramètres pour la fonction voulue
VOLUME	numéro fonction, paramètres pour la fonction voulue
ELECTRICITE	numéro fonction, paramètres pour la fonction voulue
PHYSIQUE	numéro fonction, paramètres pour la fonction voulue
MATH	numéro fonction, paramètres pour la fonction voulue

TABLEAU D'IMPLANTATION DES ROUTINES EN MEMOIRE

NOM RSX ou ROUTINE	IMPLANTATION EN MEMOIRE								CHAPITRE
	ROUTINE				RSX				
	INSTRUCTIONS		DONNEES		INSTRUCTIONS		DONNEES		
	début	fin	début	fin	début	fin	début	fin	
AFFBIN	#7000	#7033	#7034	#7046					2
AFFHEX	#7000	#7043	#7044	#704A					
AFFNUM	#7000	#70EC	#70ED	#710B					
AFENTS	#7000	#710E	#710F	#712E					
ASCII	#7000	#7042	#7043	#704E					
ASCIIS	#7000	#7069	#706A	#7072					
AFFLOT	#7000	#7212	#7213	#7256					
ENTFLO	#7000	#7099	#709A	#70A9					
DECFLO	#7000	#70E9	#70EA	#70F9					
FIND	#7000	#7095	#7096	#70BF	#6FA0	#6FC2	#6FC3	#6FCB	3
CHANGE	#7000	#7111	#7112	#7169	#6FA0	#6FDC	#6FDD	#6FE7	
TRIC	#7000	#70FD	#70FE	#717D	#6FA0	#6FCD	#6FCE	#6FD6	
TRIE	#7000	#70A3	#70A4	#711D	#6FA0	#6FCD	#6FCE	#6FD6	
TRIR	#7000	#70D1	#70D2	#7152	#6FA0	#6FCD	#6FCE	#6FD6	
COULGRAF					#6FA0	#6FC6	#6FC7	#6FD3	4
FENGRAF	#7000	#7014	#7015	#701C	#6FA0	#6FD8	#6FD9	#6FE4	
TRIANGLE	#7000	#7042	#7043	#7052	#6FA0	#6FEA	#6FEB	#6FF7	
RECTANGLE 1	#7000	#7055	#7056	#7061	#6FA0	#6FD7	#6FD8	#6FE5	
RECTANGLE 2	#7000	#70E6	#70E7	#710E	#6FA0	#6FE2	#6FE3	#6FF0	
CERCLE	#7000	#7099	#709A	#70B4	#6FA0	#6FCD	#6FCE	#6FD8	
ELLIPSE	#7000	#70FD	#70FE	#7135	#6F90	#6FE1	#6FE2	#6FED	
POLYGONE	#7000	#70C9	#70CA	#70F5	#6F90	#6FE4	#6FE5	#6FF1	
COPZONE	#7000	#7067	#7068	#7079	#6FA0	#6FEC	#6FED	#6FF8	
ZOOM	#A050	#A12A	#A12B	#A520	#A000	#A024	#A025	#A02D	
SUPERZOOM	#9A50	#9BAE	#9BAF	#9BC2	#9A00	#9A24	#9A25	#9A32	
AIRE	#8080	#81C3	#81C4	#81DF	#8000	#805C	#805D	#8065	5
VOLUME	#8080	#822F	#8230	#825A	#8000	#805C	#805D	#8067	
ELECTRICITE	#8080	#8209	#820A	#8225	#8000	#805C	#805D	#806C	
PHYSIQUE	#8080	#8217	#8218	#823A	#8000	#8074	#8075	#8081	
MATH	#8080	#820B	#820C	#823B	#8000	#805C	#805D	#8065	

TABLEAU DES LONGUEURS

NOM RSX ou ROUTINE	LONGUEUR (en décimal)							CHAPITRE
	ROUTINE			RSX			TOTAL	
	INSTR.	DONNEES	TOTAL	INSTR.	DONNEES	TOTAL		
AFFBIN	52	19	71				71	2
AFFHEX	68	7	75				75	
AFFNUM	237	31	268				268	
AFENTS	271	32	303				303	
ASCII	67	8	75				75	
ASCIIS	106	9	115				115	
AFFLOT	531	68	599				599	
ENTFLO	154	16	170				170	
DECFLO	234	16	250				250	
FIND	150	42	192	35	9	44	236	3
CHANGE	274	88	362	61	11	72	434	
TRIC	254	128	382	46	9	55	437	
TRIE	164	122	286	46	9	55	341	
TRIR	210	129	339	46	9	55	394	
COULGRAF				39	13	52	52	4
FENGRAF	21	8	29	57	12	69	98	
TRIANGLE	67	16	83	75	13	88	171	
RECTANGLE 1	86	12	98	56	14	70	168	
RECTANGLE 2	231	40	271	67	14	81	352	
CERCLE	154	27	181	46	11	57	238	
ELLIPSE	254	56	310	82	12	94	404	
POLYGONE	202	44	246	85	13	98	344	
COPZONE	104	18	122	77	12	89	211	
ZOOM	219	1014	1233	37	9	46	1279	
SUPERZOOM	351	20	371	37	14	51	422	
AIRE	324	28	352	93	9	102	454	5
VOLUME	432	43	475	93	11	104	579	
ELECTRICITE	394	28	422	93	16	109	531	
PHYSIQUE	392	35	427	117	13	130	557	
MATH	396	48	444	93	9	102	546	

ANNEXE 2

LES CHARGEURS DE ROUTINES

Dans cette annexe, vous trouverez le chargeur correspondant à la routine que vous voulez charger en mémoire.

Les chargeurs sont présentés suivant l'ordre de l'apparition de la routine dans l'ouvrage.

Nous avions dit, en introduction, que les routines pouvaient être facilement relogeables ; si vous en éprouvez le besoin, il vous sera très facile d'intégrer quelques instructions supplémentaires au CHARGEUR pour y arriver.

Pour ce faire, servez-vous du TABLEAU D'IMPLANTATION EN MEMOIRE de l'annexe 1 ; en effet, il vous faudra couper les lignes de DATA en deux parties, ou quatre parties distinctes, soit :

— instructions RSX
— données RSX
— instructions ROUTINE
— données ROUTINE

La méthode est assez simple ; il faut dans un premier temps dupliquer les lignes 10 à 70 (60 non comprise) en 2 ou 4 fois, selon qu'il s'agit d'une routine non RSX ou RSX.

La variable "d" indique l'adresse de début d'implantation, et la variable "f" l'adresse de fin d'implantation.

PRÉCAUTIONS IMPORTANTES

(1) il faut modifier les lignes DATA en fonction du nouvel emplacement des données. Ceci se fera à l'aide du listing assembleur de la routine.

(2) il faut modifier, ou mieux supprimer, le contrôle effectué à l'aide du 17e élément de chaque ligne de DATA.
La suppression est fort simple ; il vous suffit pour cela de supprimer cet élément, ainsi que les lignes 50 et 60 du CHARGEUR, où s'effectuent respectivement, la lecture et le contrôle de celui-ci.

PRÉCISION CONCERNANT LE CONTRÔLE

Le contrôle de l'exactitudes des octets des lignes se fait en comparant le 17e élément de chaque ligne de DATA avec la somme de tous les octets précédents.

Chargeur AFFBIN

```
10 d=&7000:f=&7046:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA CD,00,B9,ED,4B,34,70,DD,21,46,7
0,1E,01,16,07,79,05CB
110 DATA DD,2B,CB,3F,38,06,DD,36,00,30,1
8,04,DD,36,00,31,0ABE
120 DATA 15,CB,7A,28,EB,1D,CB,7B,20,03,4
8,18,E0,21,36,70,10B8
130 DATA CD,7D,C3,C9,3B,F9,00,00,00,00,0
0,00,00,00,00,00,14C2
140 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,14C2
```

Chargeur AFFHEX

```
10 d=&7000:f=&704A:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA CD,00,B9,ED,4B,44,70,DD,21,4A,7
0,1E,01,79,E6,0F,06B7
110 DATA FE,0A,38,02,CE,07,A7,CE,30,DD,2
B,DD,77,00,79,CB,0E13
120 DATA 3F,CB,3F,CB,3F,CB,3F,FE,0A,38,0
2,CE,07,A7,CE,30,152C
130 DATA DD,2B,DD,77,00,1D,CB,7B,20,03,4
8,18,D0,21,46,70,1B15
140 DATA CD,7D,C3,C9,E3,F0,00,00,00,00,0
0,00,00,00,00,00,1FBE
```

Chargeur AFFNUM

```
10 d=&7000:f=&710B:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA CD,00,B9,3E,05,DD,21,EF,70,DD,3
6,00,20,DD,23,3D,0696
110 DATA 20,F7,01,F4,70,ED,43,0A,71,3E,0
0,32,F4,70,3E,0A,0CD9
120 DATA DD,21,FA,70,DD,36,00,00,DD,23,3
D,20,F7,2A,ED,70,142F
130 DATA 11,FA,70,CD,64,BD,21,FA,70,11,F
5,70,01,05,00,ED,1B8C
140 DATA B0,2A,06,71,11,FF,70,CD,64,BD,2
1,FA,70,11,FF,70,2356
150 DATA CD,88,BD,21,FA,70,CD,73,BD,ED,4
B,FA,70,ED,43,08,2CCA
160 DATA 71,2A,08,71,11,FA,70,CD,64,BD,2
A,06,71,11,FF,70,3368
170 DATA CD,64,BD,18,02,18,A7,11,FF,70,2
1,FA,70,CD,85,BD,3B49
180 DATA 21,FA,70,11,FF,70,01,05,00,ED,B
0,21,F5,70,11,FA,4288
190 DATA 70,01,05,00,ED,B0,DD,21,FF,70,D
D,7E,04,FE,00,CC,4A31
200 DATA E8,70,21,FA,70,11,FF,70,CD,82,B
D,2A,04,71,11,FF,524F
210 DATA 70,CD,64,BD,21,FA,70,11,FF,70,C
D,7C,BD,21,FA,70,5B49
220 DATA CD,73,BD,2A,FA,70,ED,4B,0A,71,0
B,ED,43,0A,71,7D,62C0
230 DATA 2A,0A,71,77,2A,08,71,7D,FE,00,2
8,05,22,ED,70,18,67BE
240 DATA 94,21,EF,70,CD,7D,C3,C9,DD,36,0
4,80,C9,39,30,00,6F71
250 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,6F71
260 DATA 00,00,00,00,30,00,0A,00,00,00,0
0,00,00,00,00,00,6FAB
```

Chargeur AFENTS

```
10 d=&7000:f=&712E:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA CD,00,B9,3E,06,DD,21,11,71,DD,3
6,00,20,DD,23,3D,05BA
110 DATA 20,F7,01,17,71,ED,43,2D,71,3E,0
0,32,17,71,2A,0F,0A59
120 DATA 71,E5,CB,7C,28,0A,7C,2F,67,7D,2
F,6F,23,22,0F,71,101A
130 DATA 3E,0A,DD,21,1D,71,DD,36,00,00,D
D,23,3D,20,F7,2A,157F
140 DATA 0F,71,11,1D,71,CD,64,BD,21,1D,7
1,11,18,71,01,05,19DB
150 DATA 00,ED,B0,2A,29,71,11,22,71,CD,6
4,BD,21,1D,71,11,1F8E
160 DATA 22,71,CD,88,BD,21,1D,71,CD,73,B
D,ED,4B,1D,71,ED,2792
170 DATA 43,2B,71,2A,2B,71,11,1D,71,CD,6
4,BD,2A,29,71,11,2C99
180 DATA 22,71,CD,64,BD,18,02,18,A7,11,2
2,71,21,1D,71,CD,3213
190 DATA 85,BD,21,1D,71,11,22,71,01,05,0
0,ED,B0,21,18,71,36F5
200 DATA 11,1D,71,01,05,00,ED,B0,DD,21,2
2,71,DD,7E,04,FE,3D25
210 DATA 00,CC,0A,71,21,1D,71,11,22,71,C
D,82,BD,2A,27,71,428D
220 DATA 11,22,71,CD,64,BD,21,1D,71,11,2
2,71,CD,7C,BD,21,4899
230 DATA 1D,71,CD,73,BD,2A,1D,71,ED,4B,2
D,71,0B,ED,43,2D,4F1A
240 DATA 71,7D,2A,2D,71,77,2A,2B,71,7D,F
E,00,28,05,22,0F,53E6
250 DATA 71,18,94,E1,CB,7C,28,04,3E,2D,1
8,02,3E,2B,2A,2D,589C
260 DATA 71,2D,77,21,11,71,CD,7D,C3,C9,D
D,36,04,80,C9,D5,605F
270 DATA DD,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,613C
280 DATA 00,00,00,00,00,00,00,30,00,0A,0
0,00,00,00,00,00,6176
```

Chargeur ASCII

```
10 d=&7000:f=&704A:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA CD,00,B9,ED,4B,43,70,DD,21,45,7
0,11,10,27,CD,30,0669
110 DATA 70,11,E8,03,CD,30,70,11,64,00,C
D,30,70,11,0A,00,0B3F
120 DATA CD,30,70,11,01,00,CD,30,70,21,4
5,70,CD,7D,C3,C9,11D7
130 DATA 3E,2F,60,69,3C,ED,52,30,FB,ED,5
A,44,4D,DD,77,00,18DF
140 DATA DD,23,C9,7D,00,00,00,00,00,00,0
0,00,00,00,00,00,1B25
```

Chargeur ASCIIS

```
10 d=&7000:f=&7072:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA CD,00,B9,ED,4B,6A,70,CB,78,28,0
B,78,2F,47,79,2F,06A4
110 DATA 4F,03,3E,2D,18,02,3E,2B,32,6C,7
0,DD,21,6D,70,11,0ADE
120 DATA 10,27,CD,57,70,11,E8,03,CD,57,7
0,11,64,00,CD,57,10D2
130 DATA 70,11,0A,00,CD,57,70,11,01,00,C
D,57,70,21,6D,70,1595
140 DATA 2B,01,07,00,0B,23,7E,FE,30,28,F
9,11,6D,70,ED,B0,1B4E
150 DATA 21,6C,70,CD,7D,C3,C9,3E,2F,60,6
9,3C,ED,52,30,FB,22FD
160 DATA ED,5A,44,4D,DD,77,00,DD,23,C9,D
5,DD,00,00,00,00,29A4
170 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,29A4
```

Chargeur AFFLOT

```
10 d=&7000:f=&7256:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA CD,00,B9,DD,21,13,72,3E,00,DD,7
7,00,3E,00,DD,77,062D
110 DATA 01,3E,00,DD,77,02,3E,A8,DD,77,0
3,3E,84,DD,77,04,0C19
120 DATA DD,21,13,72,DD,7E,03,CB,7F,20,0
4,3E,2B,18,02,3E,1129
130 DATA 2D,32,56,72,DD,7E,03,CB,27,CB,2
F,DD,77,03,21,13,1725
140 DATA 72,11,40,72,01,05,00,ED,B0,21,4
0,72,CD,73,BD,2A,1CF7
150 DATA 40,72,22,18,72,2A,18,72,11,40,7
2,CD,64,BD,DD,21,22B8
160 DATA 40,72,DD,7E,03,CB,27,CB,2F,DD,7
7,03,21,40,72,11,28EF
170 DATA 45,72,01,05,00,ED,B0,21,13,72,1
1,40,72,01,05,00,2CB8
180 DATA ED,B0,21,40,72,11,45,72,CD,82,B
D,2A,4A,72,11,45,3338
190 DATA 72,CD,64,BD,21,40,72,11,45,72,C
D,85,BD,21,40,72,3A15
200 DATA CD,73,BD,2A,40,72,22,1A,72,2A,1
8,72,22,54,72,CD,4005
210 DATA 1B,71,21,1C,72,11,23,72,01,06,0
0,ED,B0,2A,1A,72,4440
220 DATA 22,54,72,CD,1B,71,21,1C,72,11,2
9,72,01,06,00,ED,48D0
230 DATA B0,01,06,00,21,23,72,11,2F,72,7
E,FE,20,20,04,23,4CD2
240 DATA 0D,18,F7,C5,ED,B0,C1,21,2F,72,0
9,3E,2C,77,23,54,5334
250 DATA 5D,21,29,72,01,06,00,7E,FE,20,2
0,02,18,0A,FE,2D,575F
260 DATA 20,02,18,04,FE,2B,20,04,23,0D,1
8,EB,ED,B0,3A,56,5C4A
270 DATA 72,32,2F,72,21,2F,72,CD,7D,C3,C
9,3E,06,DD,21,1C,6285
280 DATA 72,DD,36,00,20,DD,23,3D,20,F7,0
1,22,72,ED,43,52,6895
```

```
290 DATA 72,3E,00,32,22,72,2A,54,72,E5,3
E,0A,DD,21,40,72,6DD8
300 DATA DD,36,00,00,DD,23,3D,20,F7,2A,5
4,72,11,40,72,CD,73BF
310 DATA 64,BD,21,40,72,11,3B,72,01,05,0
0,ED,B0,2A,4E,72,78FE
320 DATA 11,45,72,CD,64,BD,21,40,72,11,4
5,72,CD,88,BD,21,7F82
330 DATA 40,72,CD,73,BD,ED,4B,40,72,ED,4
3,50,72,2A,50,72,86F9
340 DATA 11,40,72,CD,64,BD,2A,4E,72,11,4
5,72,CD,64,BD,18,8D62
350 DATA 02,18,A7,11,45,72,21,40,72,CD,8
5,BD,21,40,72,11,92B1
360 DATA 45,72,01,05,00,ED,B0,21,3B,72,1
1,40,72,01,05,00,96A2
370 DATA ED,B0,DD,21,45,72,DD,7E,04,FE,0
0,CC,0E,72,21,40,9DFE
380 DATA 72,11,45,72,CD,82,BD,2A,4C,72,1
1,45,72,CD,64,BD,A4E2
390 DATA 21,40,72,11,45,72,CD,7C,BD,21,4
0,72,CD,73,BD,2A,AB7D
400 DATA 40,72,ED,4B,52,72,0B,ED,43,52,7
2,7D,2A,52,72,77,B20C
410 DATA 2A,50,72,7D,FE,00,28,05,22,54,7
2,18,94,E1,CB,7C,B85C
420 DATA 28,04,3E,2D,18,02,3E,2B,2A,52,7
2,2D,77,C9,DD,36,BCE4
430 DATA 04,80,C9,00,00,00,00,00,00,00,0
0,00,00,00,00,00,BE31
440 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,BE31
450 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,BE31
460 DATA 00,00,00,00,00,00,00,00,00,00,E
8,03,30,00,0A,00,BF56
470 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,BF56
```

Chargeur ENTFLO

```
10 d=&7000:f=&70A9:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA CD,00,B9,DD,21,9A,70,3E,DD,DD,7
7,00,3E,24,DD,77,07B3
110 DATA 01,3E,E7,DD,77,02,3E,40,DD,77,0
3,3E,8F,DD,77,04,0E29
120 DATA CB,BF,F5,DE,08,30,16,F1,2F,3C,C
6,08,DD,4E,03,CB,15F7
130 DATA F9,CB,39,3D,20,FB,06,00,ED,43,9
F,70,C9,F1,F5,DE,1F1E
140 DATA 10,38,08,F1,21,A1,70,CD,7D,C3,C
9,F1,2F,3C,C6,10,2699
150 DATA F5,DD,21,9A,70,DD,6E,02,DD,66,0
3,CB,FC,11,00,00,2E01
160 DATA CB,3D,3D,20,FB,F1,2F,3C,C6,08,F
5,CB,3B,CB,45,28,35BE
170 DATA 04,CB,FB,18,02,CB,BB,CB,3D,3D,2
0,EF,F1,2F,3C,C6,3D9E
180 DATA 08,28,11,CB,3B,CB,44,28,04,CB,F
B,18,02,CB,BB,CB,4451
190 DATA 3C,3D,20,EF,54,ED,53,9F,70,C9,0
0,00,00,00,00,00,4945
200 DATA 00,6F,76,65,72,66,6C,6F,77,00,0
0,00,00,00,00,00,4CB9
```

Chargeur SECFLO

```
10 d=&7000:f=&70F9:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA CD,00,B9,DD,21,EA,70,3E,DD,DD,7
7,00,3E,24,DD,77,0803
110 DATA 01,3E,E7,DD,77,02,3E,40,DD,77,0
3,3E,89,DD,77,04,0E73
120 DATA CB,BF,4F,F5,DE,09,30,0C,DD,7E,0
2,6F,DD,7E,03,67,15F5
130 DATA CB,FC,18,1B,F1,F5,DE,10,38,08,F
1,21,F1,70,CD,7D,1EC0
140 DATA C3,C9,F1,F5,DE,08,4F,DD,7E,01,6
F,DD,7E,02,67,F1,27E7
150 DATA 79,F5,DE,00,28,05,CB,24,3D,20,F
B,F1,2F,3C,C6,08,2ED1
160 DATA F5,DE,06,CB,7F,28,48,F1,F5,11,0
0,00,CB,23,CB,7C,3690
170 DATA 28,04,CB,C3,18,02,CB,83,CB,24,3
D,20,EF,F1,2F,3C,3D49
180 DATA C6,06,28,15,CB,7F,20,11,CB,23,C
B,7D,28,04,CB,C3,43BD
190 DATA 18,02,CB,83,CB,25,3D,20,EF,3E,0
3,62,6B,19,29,19,48CA
200 DATA 29,54,5D,3D,20,F7,11,FA,00,19,1
9,CD,B4,70,C9,F1,4FE0
210 DATA 3E,06,18,B4,3E,06,11,00,00,CB,3
D,3D,20,FB,3E,02,53E5
220 DATA CB,3B,CB,45,28,04,CB,FB,18,02,C
B,BB,CB,3D,3D,20,5AF2
230 DATA EF,3E,06,CB,3B,CB,44,28,04,CB,F
B,18,02,CB,BB,CB,6297
240 DATA 3C,3D,20,EF,54,ED,53,EF,70,C9,0
0,00,00,00,00,00,67DB
250 DATA 00,6F,76,65,72,66,6C,6F,77,00,0
0,00,00,00,00,00,6B4F
```

Chargeur FIND

```
10 d=&6FA0:f=&70BF:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,C8,6F,CD,D1,BC,C9,C
3,6F,DD,66,05,DD,08EC
110 DATA 6E,04,DD,46,03,DD,4E,02,DD,56,0
1,DD,5E,00,FE,03,0F21
120 DATA C0,18,3D,46,49,4E,C4,00,00,00,0
0,00,00,00,00,00,11D7
130 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,11D7
140 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,11D7
150 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,11D7
160 DATA ED,53,98,70,CD,00,B9,EB,3E,00,B
E,28,22,EB,DD,2A,19C8
170 DATA 98,70,DD,7E,00,BE,20,2D,22,96,7
0,DD,23,DD,7E,00,20B9
180 DATA FE,00,28,12,CD,4B,70,28,05,BE,2
8,EF,18,11,C9,21,268E
190 DATA A1,70,CD,7D,C3,C9,E5,C5,D5,CD,5
6,70,D1,C1,E1,CD,31C7
200 DATA 4B,70,20,CA,C9,CD,4B,70,20,CB,C
9,23,E5,D1,E5,C5,3AF4
210 DATA E1,A7,ED,52,E1,C9,ED,4B,96,70,D
D,21,9A,70,11,10,43CC
220 DATA 27,CD,83,70,11,E8,03,CD,83,70,1
1,64,00,CD,83,70,4AA4
230 DATA 11,0A,00,CD,83,70,11,01,00,CD,8
3,70,21,9A,70,CD,5049
240 DATA 7D,C3,C9,3E,2F,60,69,3C,ED,52,3
0,FB,ED,5A,44,4D,5806
250 DATA DD,77,00,DD,23,C9,00,00,00,00,0
0,00,00,00,00,0A,5B2D
260 DATA 00,74,65,78,74,65,20,65,6E,74,7
2,65,20,69,6E,63,60EF
270 DATA 6F,72,72,65,63,74,20,2F,20,61,7
2,72,65,74,0A,00,6615
```

Chargeur CHANGE

```
10 d=&6FA0:f=&7169:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,E4,6F,CD,D1,BC,C9,D
D,6F,DD,66,07,DD,0924
110 DATA 6E,06,DD,46,05,DD,4E,04,ED,43,3
7,71,DD,56,03,DD,0FDA
120 DATA 5E,02,ED,53,14,71,DD,56,01,DD,5
E,00,ED,53,31,71,1650
130 DATA DD,2A,14,71,FD,2A,31,71,FE,04,C
0,18,23,43,48,41,1C6E
140 DATA 4E,47,C5,00,00,00,00,00,00,00,0
0,00,00,00,00,00,1DC8
150 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,1DC8
160 DATA CD,00,B9,11,00,00,DD,7E,00,FE,0
0,28,05,13,DD,23,22F8
170 DATA 18,F4,ED,53,35,71,7B,D6,00,28,4
1,DD,2A,14,71,11,2941
180 DATA 00,00,FD,7E,00,FE,00,28,05,13,F
D,23,18,F4,ED,53,2F66
190 DATA 33,71,7B,D6,00,28,25,FD,2A,31,7
1,DD,2A,14,71,DD,35DA
200 DATA 7E,00,BE,20,48,22,12,71,DD,23,D
D,7E,00,FE,00,28,3BA4
210 DATA 1A,CD,93,70,28,05,BE,28,EF,18,2
C,C9,21,46,71,CD,4242
220 DATA 7D,C3,C9,E1,21,16,71,CD,7D,C3,C
9,E5,C5,E1,1E,1E,4B71
230 DATA 3E,00,BE,20,EE,2B,1D,20,F9,E1,E
5,C5,D5,DD,E5,CD,54CB
240 DATA 9E,70,DD,E1,D1,C1,E1,CD,93,70,2
0,AF,C9,CD,93,70,5F42
250 DATA 20,B0,C9,23,E5,D1,E5,C5,E1,A7,E
D,52,E1,C9,21,3D,692D
260 DATA 71,CD,7D,C3,2A,33,71,ED,5B,35,7
1,A7,ED,52,E5,C5,71F7
270 DATA E1,ED,5B,12,71,A7,ED,52,ED,5B,3
5,71,ED,52,E5,2A,7AC5
```

```
280 DATA 12,71,ED,5B,35,71,A7,ED,5A,22,3
9,71,2A,12,71,ED,818A
290 DATA 5B,33,71,A7,ED,5A,22,3B,71,E1,D
1,CB,7A,20,16,A7,8919
300 DATA ED,52,E5,D1,E5,2A,39,71,A7,ED,5
A,ED,5B,37,71,C1,9266
310 DATA 03,ED,B8,18,05,CD,08,71,ED,B0,E
D,4B,33,71,2A,31,9945
320 DATA 71,ED,5B,12,71,ED,B0,C9,E5,C1,2
A,39,71,ED,5B,3B,A1E4
330 DATA 71,C9,00,00,00,00,66,69,6E,20,6
4,65,20,7A,6F,6E,A6BB
340 DATA 65,20,74,65,78,74,65,20,61,74,7
4,65,69,6E,74,0A,AC8D
350 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,63,68,61,ADB9
360 DATA 6E,67,65,20,0A,00,6C,6F,6E,67,7
5,65,75,72,20,74,B322
370 DATA 65,78,74,65,28,73,29,20,6E,75,6
C,20,2D,20,66,69,B847
380 DATA 6E,20,74,72,61,69,74,2E,0A,00,0
0,00,00,00,00,00,BB31
```

Chargeur TRIC

```
10 d=&6FA0:f=&717D:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,D3,6F,CD,D1,BC,C9,C
E,6F,FE,01,28,05,0809
110 DATA FE,02,28,0C,C9,01,00,00,DD,66,0
1,DD,6E,00,18,40,0CEE
120 DATA DD,66,03,DD,6E,02,DD,46,01,DD,4
E,00,18,32,54,52,12C0
130 DATA 49,C3,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,13CC
140 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,13CC
150 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,13CC
160 DATA CD,00,B9,2B,2B,E5,DD,E1,0B,CB,7
8,28,08,DD,46,01,1AED
170 DATA DD,4E,00,18,1A,03,CB,78,28,15,2
1,57,71,CD,7D,C3,20C3
180 DATA C9,21,08,71,CD,7D,C3,C9,21,2F,7
1,CD,7D,C3,C9,2B,28BE
190 DATA 3E,01,BE,20,EC,2B,2B,2B,3E,02,B
E,20,EB,DD,2B,DD,2F36
200 DATA 22,06,71,ED,43,FE,70,FD,21,01,0
0,DD,23,DD,23,DD,3669
210 DATA 23,DD,7E,03,32,04,71,DD,7E,00,3
2,02,71,DD,E5,DD,3D30
220 DATA 6E,04,DD,66,05,DD,5E,01,DD,56,0
2,D5,DD,E1,DD,7E,4549
230 DATA 00,BE,28,09,5E,A7,93,CB,7F,20,5
D,18,2B,ED,5B,02,4B24
240 DATA 71,1B,ED,53,02,71,CB,7A,20,4E,E
D,5B,04,71,1B,ED,51DB
250 DATA 53,04,71,CB,7A,20,11,DD,23,23,1
8,D2,FD,E1,18,AB,58C7
260 DATA FD,E1,DD,2A,06,71,18,9F,DD,E1,D
D,E5,DD,56,02,DD,626C
270 DATA 5E,01,ED,53,00,71,DD,56,05,DD,5
E,04,DD,72,02,DD,6921
```

```
280 DATA 73,01,ED,5B,00,71,DD,72,05,DD,7
3,04,DD,7E,00,DD,702E
290 DATA 5E,03,DD,73,00,DD,77,03,DD,E1,F
D,E5,D1,13,D5,2A,78B9
300 DATA FE,70,2B,A7,ED,52,CB,7C,28,B2,2
A,FE,70,2B,22,FE,813C
310 DATA 70,11,02,00,A7,ED,52,CB,7C,28,A
5,FD,E1,C9,00,00,8860
320 DATA 00,00,00,00,00,00,00,00,6C,65,2
0,74,61,62,6C,65,8B59
330 DATA 61,75,20,61,20,70,6C,75,73,20,6
4,27,75,6E,20,69,90AB
340 DATA 6E,64,69,63,65,20,2F,20,61,72,7
2,65,74,0A,00,6C,95B1
350 DATA 65,20,74,61,62,6C,65,61,75,20,6
E,27,65,73,74,20,9B35
360 DATA 70,61,73,20,65,6E,20,27,6E,6F,6
D,24,27,20,2F,20,9FB7
370 DATA 61,72,72,65,74,0A,00,6E,62,20,6
5,6C,65,6D,65,6E,A545
380 DATA 74,73,20,61,20,74,72,69,65,72,2
0,69,6E,63,6F,72,AB2E
390 DATA 72,65,63,74,20,2F,20,61,72,72,6
5,74,0A,00,00,00,AF73
```

Chargeur TRIE

```
10 d=&6FA0:f=&711D:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,D3,6F,CD,D1,BC,C9,C
E,6F,FE,01,28,05,0809
110 DATA FE,02,28,0C,C9,01,00,00,DD,66,0
1,DD,6E,00,18,40,0CEE
120 DATA DD,66,03,DD,6E,02,DD,46,01,DD,4
E,00,18,32,54,52,12C0
130 DATA 49,C5,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,13CE
140 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,13CE
150 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,13CE
160 DATA CD,00,B9,2B,2B,E5,DD,E1,0B,CB,7
8,28,08,DD,46,01,1AEF
170 DATA DD,4E,00,18,1A,03,CB,78,28,15,2
1,F7,70,CD,7D,C3,2164
180 DATA C9,21,A8,70,CD,7D,C3,C9,21,CF,7
0,CD,7D,C3,C9,2B,2A9D
190 DATA 3E,01,BE,20,EC,2B,2B,2B,3E,01,B
E,20,EB,DD,22,A6,30D4
200 DATA 70,ED,43,A4,70,FD,21,01,00,DD,2
3,DD,23,DD,66,01,37EB
210 DATA DD,6E,00,DD,56,03,DD,5E,02,E5,D
5,A7,ED,52,CB,7C,4090
220 DATA 20,1C,D1,E1,DD,72,01,DD,73,00,D
D,74,03,DD,75,02,47C6
230 DATA 18,0E,FD,E1,18,D3,FD,E1,DD,2A,A
6,70,18,C7,D1,E1,5141
240 DATA FD,E5,D1,13,D5,2A,A4,70,2B,A7,E
D,52,CB,7C,28,E2,5A7C
250 DATA 2A,A4,70,2B,22,A4,70,11,02,00,A
7,ED,52,CB,7C,28,6083
260 DATA D5,FD,E1,C9,00,00,00,00,6C,65,2
0,74,61,62,6C,65,66F8
270 DATA 61,75,20,61,20,70,6C,75,73,20,6
4,27,75,6E,20,69,6C4A
```

```
280 DATA 6E,64,69,63,65,20,2F,20,61,72,7
2,65,74,0A,00,6C,7150
290 DATA 65,20,74,61,62,6C,65,61,75,20,6
E,27,65,73,74,20,76D4
300 DATA 70,61,73,20,65,6E,20,27,6E,6F,6
D,25,27,20,2F,20,7B57
310 DATA 61,72,72,65,74,0A,00,6E,62,20,6
5,6C,65,6D,65,6E,80E5
320 DATA 74,73,20,61,20,74,72,69,65,72,2
0,69,6E,63,6F,72,86CE
330 DATA 72,65,63,74,20,2F,20,61,72,72,6
5,74,0A,00,00,00,8B13
```

Chargeur TRIR

```
10 d=&6FA0:f=&7152:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,D3,6F,CD,D1,BC,C9,C
E,6F,FE,01,28,05,0809
110 DATA FE,02,28,0C,C9,01,00,00,DD,66,0
1,DD,6E,00,18,40,0CEE
120 DATA DD,66,03,DD,6E,02,DD,46,01,DD,4
E,00,18,32,54,52,12C0
130 DATA 49,D2,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,13DB
140 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,13DB
150 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,13DB
160 DATA CD,00,B9,2B,2B,E5,DD,E1,0B,CB,7
8,28,08,DD,46,01,1AFC
170 DATA DD,4E,00,18,1A,03,CB,78,28,15,2
1,2C,71,CD,7D,C3,20A7
180 DATA C9,21,E0,70,CD,7D,C3,C9,21,07,7
1,CD,7D,C3,C9,2B,2951
190 DATA 3E,01,BE,20,EC,2B,2B,2B,3E,04,B
E,20,EB,DD,2B,DD,2FCB
200 DATA 2B,DD,2B,DD,22,D4,70,ED,43,D2,7
0,FD,21,01,00,DD,37AF
210 DATA 23,DD,23,DD,23,DD,23,DD,23,FD,E
5,DD,E5,DD,E5,E1,4219
220 DATA 11,D6,70,01,05,00,ED,B0,DD,E5,E
1,23,23,23,23,23,4865
230 DATA 11,DB,70,01,05,00,ED,B0,21,D6,7
0,11,DB,70,CD,8E,4F82
240 DATA BD,FE,FF,28,29,21,DB,70,D1,D5,0
1,05,00,ED,B0,21,5763
250 DATA D6,70,D1,D5,13,13,13,13,13,01,0
5,00,ED,B0,D1,D5,5DF7
260 DATA 18,0C,FD,E1,18,A9,FD,E1,DD,2A,D
4,70,18,9D,DD,E1,6756
270 DATA D1,13,D5,2A,D2,70,2B,A7,ED,52,C
B,7C,28,E4,2A,D2,6FDB
```

```
280 DATA 70,2B,22,D2,70,11,02,00,A7,ED,5
2,CB,7C,28,D7,FD,7716
290 DATA E1,C9,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,78C0
300 DATA 6C,65,20,74,61,62,6C,65,61,75,2
0,61,20,70,6C,75,7E81
310 DATA 73,20,64,27,75,6E,20,69,6E,64,6
9,63,65,20,2F,20,837D
320 DATA 61,72,72,65,74,0A,00,6C,65,20,7
4,61,62,6C,65,61,88FF
330 DATA 75,20,6E,27,65,73,74,20,70,61,7
3,20,65,6E,20,27,8E13
340 DATA 6E,6F,6D,27,20,2F,20,61,72,72,6
5,74,6E,62,20,65,9366
350 DATA 6C,65,6D,65,6E,74,73,20,61,20,7
4,72,69,65,72,20,9945
360 DATA 69,6E,63,6F,72,72,65,63,74,20,2
F,20,61,72,72,65,9F27
370 DATA 74,0A,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,9FA5
```

Chargeur COULGRAF

```
10 d=&6FA0:f=&6FD3:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,D0,6F,CD,D1,BC,C9,C
7,6F,FE,02,28,0B,0806
110 DATA FE,01,C0,DD,4E,00,79,CD,DE,BB,C
9,DD,5E,00,7B,CD,111B
120 DATA E4,BB,DD,4E,02,18,EF,43,4F,55,4
C,47,52,41,C6,00,17C1
130 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,17C1
```

Chargeur FENGRAF

```
10 d=&6FA0:f=&701C:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,E1,6F,CD,D1,BC,C9,D
9,6F,DD,46,07,DD,08FD
110 DATA 4E,06,ED,43,15,70,DD,46,05,DD,4
E,04,ED,43,17,70,0F14
120 DATA DD,46,03,DD,4E,02,ED,43,19,70,D
D,46,01,DD,4E,00,156F
130 DATA ED,43,1B,70,FE,04,C0,18,27,46,4
5,4E,47,52,41,C6,1BA4
140 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,1BA4
150 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,1BA4
160 DATA ED,5B,15,70,2A,17,70,CD,CF,BB,E
D,5B,19,70,2A,1B,228F
170 DATA 70,CD,D2,BB,C9,00,00,00,00,00,0
0,00,00,00,00,00,2622
```

Chargeur TRIANGLE

```
10 d=&6FA0:f=&7052:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,F4,6F,CD,D1,BC,C9,E
B,6F,DD,46,01,DD,091C
110 DATA 4E,00,ED,43,4D,70,DD,56,03,DD,5
E,02,ED,53,4B,70,0FC5
120 DATA DD,66,05,DD,6E,04,22,49,70,DD,4
6,07,DD,4E,06,ED,167F
130 DATA 43,47,70,DD,66,09,DD,6E,08,22,4
5,70,DD,46,0B,DD,1CFA
140 DATA 4E,0A,ED,43,43,70,FE,06,C0,18,1
5,54,52,49,41,4E,22A4
150 DATA 47,4C,C5,00,00,00,00,00,00,00,0
0,00,00,00,00,00,23FC
160 DATA CD,CC,BB,ED,53,4F,70,22,51,70,1
1,00,00,21,00,00,2964
170 DATA CD,C9,BB,ED,5B,43,70,2A,45,70,D
5,E5,CD,EA,BB,ED,33A8
180 DATA 5B,47,70,2A,49,70,CD,F6,BB,ED,5
B,4B,70,2A,4D,70,3B05
190 DATA CD,F6,BB,E1,D1,CD,F6,BB,ED,5B,4
F,70,2A,51,70,CD,4572
200 DATA C9,BB,C9,00,00,00,00,00,00,00,0
0,00,00,00,00,00,47BF
210 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,47BF
```

Chargeur RECTANG-1

```
10 d=&6FA0:f=&7061:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,E2,6F,CD,D1,BC,C9,D
8,6F,DD,46,01,DD,08F7
110 DATA 4E,00,ED,43,58,70,DD,56,03,DD,5
E,02,ED,53,56,70,0FB6
120 DATA DD,66,05,DD,6E,04,22,5C,70,DD,4
6,07,DD,4E,06,ED,1683
130 DATA 43,5A,70,FE,04,C0,18,28,52,45,4
3,54,41,4E,47,4C,1BE2
140 DATA C5,00,00,00,00,00,FE,06,C0,18,1
5,54,52,49,41,4E,2016
150 DATA 47,4C,C5,00,00,00,00,00,00,00,0
0,00,00,00,00,00,216E
160 DATA CD,CC,BB,ED,53,5E,70,22,60,70,1
1,00,00,21,00,00,26F4
170 DATA CD,C9,BB,ED,5B,5A,70,2A,5C,70,C
D,EA,BB,ED,5B,56,305D
180 DATA 70,21,00,00,CD,F9,BB,11,00,00,2
A,58,70,CD,F9,BB,36F3
190 DATA 2A,56,70,7C,2F,67,7D,2F,6F,23,E
B,21,00,00,CD,F9,3D05
200 DATA BB,ED,5B,5A,70,2A,5C,70,CD,F6,B
B,ED,5B,5E,70,2A,4586
210 DATA 60,70,CD,C9,BB,C9,00,00,00,00,0
0,00,00,00,00,00,4970
220 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,4970
```

Chargeur RECTANG-2

```
10 d=&6FA0:f=&710E:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,ED,6F,CD,D1,BC,C9,E
3,6F,DD,56,01,DD,091D
110 DATA 5E,00,ED,53,E7,70,DD,46,03,DD,4
E,02,ED,43,FB,70,1100
120 DATA DD,46,05,DD,4E,04,ED,43,F1,70,D
D,46,07,DD,4E,06,1843
130 DATA ED,43,EF,70,DD,46,09,DD,4E,08,E
D,43,ED,70,FE,05,20C1
140 DATA C0,18,1D,52,45,43,54,41,4E,47,4
C,C5,00,00,00,00,24CB
150 DATA 00,4C,C5,00,00,00,00,00,00,00,0
0,00,00,00,00,00,25DC
160 DATA CD,CC,BB,ED,53,E9,70,22,EB,70,E
D,5B,ED,70,2A,EF,3004
170 DATA 70,CD,C9,BB,2A,E7,70,11,05,71,D
5,CD,64,BD,E1,11,3882
180 DATA 0A,71,01,05,00,ED,B0,3E,FF,32,1
3,B1,21,F1,70,11,3E66
190 DATA F6,70,CD,C7,70,21,FB,70,11,00,7
1,CD,C7,70,21,05,4608
200 DATA 71,CD,AF,BD,11,F1,70,EB,CD,85,B
D,21,FB,70,11,05,4EC0
210 DATA 71,CD,85,BD,21,0A,71,CD,AC,BD,1
1,F6,70,EB,CD,85,57C6
220 DATA BD,21,00,71,11,0A,71,CD,85,BD,2
1,F1,70,CD,D1,70,5F40
230 DATA 21,F6,70,CD,D1,70,21,FB,70,CD,D
1,70,21,00,71,CD,67CE
240 DATA D1,70,11,00,00,21,00,00,CD,EA,B
B,ED,5B,F1,70,2A,6E86
250 DATA F6,70,CD,F9,BB,2A,00,71,CD,DF,7
0,ED,5B,FB,70,EB,78C2
```

```
260 DATA CD,F9,BB,2A,F1,70,CD,DF,70,EB,2
A,F6,70,CD,DF,70,8381
270 DATA CD,F9,BB,11,00,00,21,00,00,CD,F
6,BB,ED,5B,E9,70,8B53
280 DATA 2A,EB,70,CD,C9,BB,C9,3E,00,CD,6
7,BD,EB,CD,61,BD,94F7
290 DATA C9,CD,94,BD,F5,CD,6D,BD,F1,FE,0
2,D4,DF,70,C9,7C,A023
300 DATA 2F,67,7D,2F,6F,23,C9,00,00,00,0
0,00,00,00,00,00,A2C0
310 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,A2C0
320 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,A2C0
```

Chargeur CERCLE

```
10 d=&6FA0:f=&70B4:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,D5,6F,CD,D1,BC,C9,C
E,6F,DD,46,01,DD,08E0
110 DATA 4E,00,ED,43,9A,70,DD,56,03,DD,5
E,02,ED,53,B3,70,103E
120 DATA DD,66,05,DD,6E,04,22,B1,70,FE,0
3,C0,18,32,43,45,16AB
130 DATA 52,43,4C,C5,00,00,00,00,00,08,E
D,43,ED,70,FE,05,1BE9
140 DATA C0,18,1D,52,45,43,54,41,4E,47,4
C,C5,00,00,00,00,1FF3
150 DATA 00,4C,C5,00,00,00,00,00,00,00,0
0,00,00,00,00,00,2104
160 DATA CD,CC,BB,ED,53,AB,70,22,AD,70,E
D,5B,B1,70,2A,B3,2A38
170 DATA 70,CD,C9,BB,2A,9A,70,11,A6,70,C
D,64,BD,3E,FF,32,32B1
180 DATA 13,B1,11,00,00,ED,53,AF,70,2A,A
F,70,11,9C,70,CD,3918
190 DATA 64,BD,11,A1,70,01,05,00,ED,B0,2
1,9C,70,CD,AF,BD,4064
200 DATA 11,A6,70,CD,85,BD,CD,94,BD,F5,C
D,6A,BD,F1,FE,02,4A92
210 DATA D4,92,70,22,9C,70,21,A1,70,CD,A
C,BD,11,A6,70,CD,52F2
220 DATA 85,BD,CD,6A,BD,22,A1,70,ED,5B,9
C,70,D5,E5,CD,EA,5D20
230 DATA BB,E1,D1,CD,92,70,CD,EA,BB,2A,A
F,70,23,22,AF,70,667B
240 DATA 11,B5,00,ED,52,20,A2,ED,5B,AB,7
0,2A,AD,70,CD,C9,6E82
250 DATA BB,C9,7C,2F,67,7D,2F,6F,23,C9,0
0,00,00,00,00,00,731F
260 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,731F
270 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,731F
```

Chargeur ELLIPSE

```
10 d=&6F90:f=&7135:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,9A,6F,21,EA,6F,CD,D1,BC,C9,E
2,6F,FE,05,20,10,082B
110 DATA DD,46,01,DD,4E,00,ED,43,2A,71,D
D,23,DD,23,18,08,0E65
120 DATA FE,04,C0,21,2A,71,36,00,DD,56,0
5,DD,5E,04,DD,46,14B3
130 DATA 07,DD,4E,06,ED,53,28,71,ED,43,2
6,71,DD,56,01,DD,1B9C
140 DATA 5E,00,DD,46,03,DD,4E,02,ED,43,F
E,70,ED,53,00,71,229C
150 DATA 18,1E,45,4C,4C,49,50,53,C5,00,0
0,00,00,00,00,00,2560
160 DATA 00,4C,C5,00,00,00,00,00,00,00,0
0,00,00,00,00,00,2671
170 DATA CD,CC,BB,ED,53,20,71,22,22,71,E
D,5B,26,71,2A,28,2D7C
180 DATA 71,CD,C9,BB,3E,FF,32,13,B1,2A,2
A,71,E5,11,31,71,34CE
190 DATA AF,CD,64,BD,E1,11,2C,71,AF,CD,6
4,BD,21,31,71,CD,3D27
200 DATA AC,BD,21,2C,71,CD,AF,BD,2A,FE,7
0,11,16,71,CD,64,44E8
210 DATA BD,2A,00,71,11,1B,71,CD,64,BD,1
1,00,00,ED,53,24,4A40
220 DATA 71,3E,00,2A,24,71,11,02,71,CD,6
4,BD,11,07,71,01,4EAA
230 DATA 05,00,ED,B0,21,02,71,CD,AF,BD,1
1,16,71,CD,85,BD,55C0
240 DATA 11,0C,71,EB,CD,61,BD,18,02,18,D
6,21,07,71,CD,AC,5C3E
250 DATA BD,11,1B,71,CD,85,BD,11,11,71,E
B,CD,61,BD,21,02,6333
260 DATA 71,11,2C,71,CD,85,BD,21,0C,71,1
1,31,71,CD,85,BD,69C1
```

```
270 DATA 21,07,71,11,31,71,CD,85,BD,21,1
1,71,11,2C,71,CD,6F3A
280 DATA 85,BD,21,02,71,11,07,71,CD,7C,B
D,CD,E8,70,E5,21,76CA
290 DATA 0C,71,11,11,71,CD,82,BD,CD,E8,7
0,D1,CD,EA,BB,2A,7F78
300 DATA 24,71,23,22,24,71,11,69,01,ED,5
2,20,9C,ED,5B,20,84C5
310 DATA 71,2A,22,71,CD,C9,BB,C9,CD,94,B
D,F5,CD,6A,BD,F1,8F05
320 DATA FE,02,D4,F6,70,C9,7C,2F,67,7D,2
F,6F,23,C9,00,00,9621
330 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,9621
340 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,9621
350 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,9621
360 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,9621
```

Chargeur POLYGONE

```
10 d=&6F90:f=&70F5:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,9A,6F,21,EE,6F,CD,D1,BC,C9,E
5,6F,FE,05,20,10,0832
110 DATA DD,46,01,DD,4E,00,ED,43,DD,70,D
D,23,DD,23,18,08,0F1E
120 DATA FE,04,C0,21,DD,70,36,00,DD,46,0
1,DD,4E,00,ED,43,1603
130 DATA F2,70,DD,56,03,DD,5E,02,ED,53,E
4,70,DD,66,05,DD,1E91
140 DATA 6E,04,22,E2,70,DD,46,07,DD,4E,0
6,ED,43,F0,70,ED,264F
150 DATA 43,F4,70,18,1B,50,4F,4C,59,47,4
F,4E,C5,00,00,00,2B16
160 DATA 00,00,C5,00,00,00,00,00,00,00,0
0,00,00,00,00,00,2BDB
170 DATA CD,CC,BB,ED,53,D9,70,22,DB,70,E
D,5B,E2,70,2A,E4,35CD
180 DATA 70,CD,C9,BB,3E,00,2A,F2,70,11,D
4,70,CD,64,BD,3E,3DD9
190 DATA FF,32,13,B1,2A,F0,70,11,E6,70,C
D,64,BD,21,68,01,4537
200 DATA 11,EB,70,CD,64,BD,21,EB,70,11,E
6,70,CD,88,BD,2A,4DB0
210 DATA DD,70,11,DD,70,CD,64,BD,18,09,2
1,DD,70,11,EB,70,5544
220 DATA CD,7C,BD,11,DD,70,21,CA,70,CD,6
1,BD,11,CF,70,01,5D3F
230 DATA 05,00,ED,B0,21,CA,70,CD,AF,BD,C
D,B1,70,22,CA,70,65BF
240 DATA 21,CF,70,CD,AC,BD,CD,B1,70,22,C
F,70,3A,F0,70,21,6E5F
250 DATA F4,70,BE,20,0C,2A,CF,70,ED,5B,C
A,70,CD,EA,BB,18,7722
```

```
260 DATA 0A,2A,CF,70,ED,5B,CA,70,CD,F6,B
B,3A,F4,70,3D,32,7FA2
270 DATA F4,70,FE,FF,20,A4,ED,5B,D9,70,2
A,DB,70,CD,C9,BB,8A1E
280 DATA C9,11,D4,70,CD,85,BD,CD,94,BD,F
5,CD,6A,BD,F1,FE,9541
290 DATA 02,D8,7C,2F,67,7D,2F,6F,23,C9,0
0,00,00,00,00,00,9934
300 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,9934
310 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,9934
320 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,9934
```

Chargeur COPZONE

```
10 d=&6FA0:f=&7079:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,AA,6F,21,F5,6F,CD,D1,BC,C9,E
D,6F,DD,46,0B,DD,0929
110 DATA 4E,0A,ED,43,68,70,DD,46,09,DD,4
E,08,ED,43,6A,70,0FF2
120 DATA DD,46,07,DD,4E,06,ED,43,6C,70,D
D,46,05,DD,4E,04,16B0
130 DATA ED,43,6E,70,DD,46,03,DD,4E,02,E
D,43,70,70,DD,46,1E44
140 DATA 01,DD,4E,00,ED,43,72,70,FE,06,C
0,18,13,43,4F,50,2453
150 DATA 5A,4F,4E,C5,00,00,00,00,00,00,0
0,00,00,00,00,00,260F
160 DATA 21,00,00,CD,05,BC,ED,5B,68,70,2
A,6A,70,CD,1D,BC,2C88
170 DATA 22,74,70,ED,5B,70,70,2A,72,70,C
D,1D,BC,22,76,70,3370
180 DATA ED,4B,68,70,2A,6C,70,09,ED,5B,6
A,70,EB,CD,1D,BC,3B42
190 DATA ED,5B,74,70,ED,52,22,78,70,2A,7
4,70,ED,5B,76,70,42F3
200 DATA ED,4B,78,70,ED,B0,2A,6E,70,2B,2
2,6E,70,21,6E,70,49E2
210 DATA 3E,00,BE,C8,2A,74,70,CD,29,BC,2
2,74,70,2A,76,70,507C
220 DATA CD,29,BC,22,76,70,18,D1,00,00,0
0,00,00,00,00,00,541F
230 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,541F
```

Chargeur ZOOM

```
10 d=&9000:f=&912F:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,0A,90,21,2A,90,CD,D1,BC,C9,2
5,90,DD,46,01,DD,074F
110 DATA 4E,00,ED,43,2D,91,DD,56,03,DD,5
E,02,ED,53,2B,91,0DFA
120 DATA FE,02,C0,18,2B,5A,4F,4F,CD,00,0
0,00,00,00,00,00,11C2
130 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,11C2
140 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,11C2
150 DATA CD,11,BC,FE,00,28,10,01,02,00,E
D,43,1D,95,01,28,16A0
160 DATA 00,ED,43,1F,95,18,0E,01,04,00,E
D,43,1D,95,01,14,1AA6
170 DATA 00,ED,43,1F,95,DD,21,33,91,2A,2
D,91,22,31,91,3E,2056
180 DATA 00,F5,3E,00,ED,5B,2B,91,ED,53,2
F,91,CD,DD,90,ED,28B4
190 DATA 5B,31,91,13,13,ED,53,31,91,F1,3
C,FE,19,28,03,F5,2F5D
200 DATA 18,E0,21,00,C0,11,00,40,01,00,4
0,ED,B0,CD,06,BB,34F3
210 DATA DD,21,33,91,3E,18,32,1C,95,3E,0
0,32,1B,95,CD,01,39DC
220 DATA 91,3A,1C,95,FE,00,28,06,3D,32,1
C,95,18,EB,CD,06,3F7A
230 DATA BB,21,00,40,11,00,C0,01,00,40,E
D,B0,C9,3C,F5,ED,462C
240 DATA 5B,2F,91,2A,31,91,CD,F0,BB,DD,7
7,00,DD,23,F1,21,4E11
250 DATA 1F,95,BE,C8,2A,2F,91,ED,4B,1D,9
5,09,22,2F,91,18,5422
260 DATA DC,3C,F5,DD,7E,00,CD,90,BB,DD,2
3,3A,1B,95,32,27,5BE5
270 DATA B7,3A,1C,95,32,26,B7,3E,8F,CD,5
A,BB,F1,21,1F,95,630B
280 DATA BE,C8,3A,1B,95,3C,32,1B,95,18,D
6,00,00,00,00,00,6787
```

Chargeur SUPER ZOOM

```
10 d=&9A00:f=&9BC2:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,0A,9A,21,2F,9A,CD,D1,BC,C9,2
5,9A,DD,46,01,DD,0772
110 DATA 4E,00,ED,43,B1,9B,DD,56,03,DD,5
E,02,ED,53,AF,9B,0F39
120 DATA FE,02,C0,18,2B,53,55,50,45,52,5
A,4F,4F,CD,00,00,1490
130 DATA 00,00,00,31,3A,64,65,66,73,09,3
2,09,3B,5A,4F,4E,1813
140 DATA 45,20,44,45,20,43,41,4C,43,55,4
C,20,0D,52,08,73,1BCF
150 DATA CD,11,BC,FE,00,28,17,01,02,00,E
D,43,BD,9B,01,27,2159
160 DATA 00,ED,43,BF,9B,11,8C,00,ED,53,C
1,9B,18,15,01,04,274E
170 DATA 00,ED,43,BD,9B,01,13,00,ED,43,B
F,9B,11,46,00,ED,2DB8
180 DATA 53,C1,9B,21,00,C0,11,00,59,01,0
0,40,ED,B0,CD,14,3371
190 DATA BC,ED,5B,C1,9B,ED,53,B7,9B,21,B
1,00,22,B9,9B,ED,3C98
200 DATA 5B,AF,9B,ED,53,B3,9B,2A,B1,9B,2
2,B5,9B,3E,01,F5,44E7
210 DATA ED,5B,B3,9B,CD,1D,BC,11,00,C0,A
7,ED,52,11,00,59,4C44
220 DATA 19,E5,ED,5B,B7,9B,2A,B9,9B,CD,1
D,BC,D1,EB,01,0A,54C7
230 DATA 00,ED,B0,2A,B9,9B,23,22,B9,9B,2
A,B5,9B,23,22,B5,5BEF
240 DATA 9B,F1,3C,FE,17,20,C8,18,00,11,1
8,00,ED,53,B9,9B,6289
250 DATA 21,62,01,18,02,18,97,22,B5,9B,E
5,3E,00,32,B7,9B,67EF
260 DATA 11,18,01,ED,53,B3,9B,CD,78,9B,2
1,B9,9B,3E,03,BE,6EFB
270 DATA 28,06,35,E1,23,23,18,DF,D1,CD,0
0,BB,CD,06,BB,FE,7661
```

```
280 DATA 65,CC,6C,9B,FE,73,28,44,FE,F3,2
8,0E,FE,F1,28,16,7ECA
290 DATA FE,F2,28,2A,FE,F0,28,1A,18,E2,2
A,AF,9B,11,08,00,85C3
300 DATA 19,22,AF,9B,18,AF,2A,B1,9B,2B,2
B,2B,2B,22,B1,9B,8B9F
310 DATA 18,A3,2A,B1,9B,23,23,23,23,22,B
1,9B,18,97,2A,AF,9152
320 DATA 9B,11,08,00,A7,ED,52,22,AF,9B,1
8,89,21,00,59,11,9684
330 DATA 00,C0,01,00,40,ED,B0,C9,2A,B5,9
B,ED,5B,B3,9B,CD,9EC8
340 DATA F0,BB,CD,90,BB,3A,B7,9B,32,27,B
7,3A,B9,9B,32,26,A70D
350 DATA B7,3E,8F,CD,5A,BB,3A,B7,9B,21,B
F,9B,BE,C8,3C,32,AF6E
360 DATA B7,9B,ED,5B,B3,9B,2A,BD,9B,19,2
2,B3,9B,18,C9,00,B742
370 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,B742
380 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,B742
```

Chargeur AIRE

```
10 d=&8000:f=&81DF:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,0A,80,21,62,80,CD,D1,BC,C9,5
D,80,F5,FE,04,20,07A5
110 DATA 14,DD,66,01,DD,6E,00,22,C8,81,1
1,D4,81,CD,B1,81,0F18
120 DATA DD,23,DD,23,3D,FE,03,20,14,DD,6
6,01,DD,6E,00,22,153B
130 DATA C6,81,11,CF,81,CD,B1,81,DD,23,D
D,23,3D,FE,02,20,1D3F
140 DATA 1A,DD,66,01,DD,6E,00,22,C4,81,1
1,CA,81,CD,B1,81,24AA
150 DATA DD,66,03,DD,6E,02,22,DE,81,18,2
5,F1,C9,41,49,52,2B91
160 DATA C5,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,2C56
170 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,2C56
180 DATA 3E,01,21,DE,81,BE,20,24,F1,FE,0
3,C0,21,CA,81,11,3346
190 DATA CF,81,CD,85,BD,AF,21,02,00,11,D
4,81,CD,64,BD,21,3AEC
200 DATA CA,81,11,D4,81,CD,88,BD,CD,B7,8
1,C9,3E,02,21,DE,43BC
210 DATA 81,BE,20,02,18,D2,3E,03,21,DE,8
1,BE,20,27,F1,FE,4ABC
220 DATA 04,C0,AF,21,02,00,11,D9,81,CD,6
4,BD,21,CA,81,11,5128
230 DATA CF,81,CD,7C,BD,11,D4,81,CD,85,B
D,11,D9,81,CD,88,5AB3
240 DATA BD,CD,B7,81,C9,3E,04,21,DE,81,B
E,20,1D,F1,FE,02,62EC
250 DATA C0,21,CF,81,CD,9A,BD,21,CA,81,1
1,CA,81,CD,85,BD,6C18
260 DATA 11,CF,81,CD,85,BD,CD,B7,81,C9,3
E,05,21,DE,81,BE,74D7
270 DATA 20,2D,F1,FE,02,C0,21,CF,81,CD,9
A,BD,AF,21,04,00,7C3E
280 DATA 11,D4,81,CD,64,BD,21,CA,81,11,C
A,81,CD,85,BD,11,847A
```

```
290 DATA CF,81,CD,85,BD,11,D4,81,CD,85,B
D,CD,B7,81,C9,3E,8E5A
300 DATA 06,21,DE,81,BE,20,1D,F1,FE,03,C
0,21,D4,81,CD,9A,966A
310 DATA BD,21,CA,81,11,CF,81,CD,85,BD,1
1,D4,81,CD,85,BD,9F78
320 DATA CD,B7,81,C9,3E,07,21,DE,81,BE,2
0,1D,F1,FE,03,C0,A7B8
330 DATA 21,D4,81,CD,9A,BD,21,CA,81,11,C
F,81,CD,85,BD,11,B03F
340 DATA D4,81,CD,85,BD,CD,B7,81,C9,3E,0
8,21,DE,81,BE,C0,B9B5
350 DATA F1,FE,03,C0,21,D4,81,CD,9A,BD,1
1,D4,81,CD,7C,BD,C36D
360 DATA 11,CA,81,EB,CD,85,BD,11,CF,81,C
D,85,BD,CD,B7,81,CD38
370 DATA C9,01,05,00,ED,B0,C9,21,CA,81,E
D,5B,C4,81,01,05,D46C
380 DATA 00,ED,B0,C9,00,00,00,00,00,00,0
0,00,00,00,00,00,D6D2
390 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,D6D2
```

Chargeur VOLUME

```
10 d=&8000:f=&825A:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,0A,80,21,64,80,CD,D1,BC,C9,5
D,80,F5,FE,04,20,07A7
110 DATA 14,DD,66,01,DD,6E,00,22,34,82,1
1,40,82,CD,1D,82,0D61
120 DATA DD,23,DD,23,3D,FE,03,20,14,DD,6
6,01,DD,6E,00,22,1384
130 DATA 32,82,11,3B,82,CD,1D,82,DD,23,D
D,23,3D,FE,02,20,19CF
140 DATA 1A,DD,66,01,DD,6E,00,22,30,82,1
1,36,82,CD,1D,82,1F81
150 DATA DD,66,03,DD,6E,02,22,59,82,18,2
5,F1,C9,56,4F,4C,25F9
160 DATA 55,4D,C5,00,00,00,00,00,00,00,0
0,00,00,00,00,00,2760
170 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,2760
180 DATA 3E,01,21,59,82,BE,20,20,F1,FE,0
2,C0,11,36,82,21,2D34
190 DATA 3B,82,CD,61,BD,21,36,82,11,36,8
2,CD,85,BD,11,3B,33D9
200 DATA 82,CD,85,BD,CD,23,82,C9,3E,02,2
1,59,82,BE,20,3D,3AFC
210 DATA F1,FE,02,C0,AF,21,04,00,11,3B,8
2,CD,64,BD,AF,21,420D
220 DATA 03,00,11,40,82,CD,64,BD,21,45,8
2,CD,9A,BD,21,36,4834
230 DATA 82,11,40,82,CD,A0,BD,11,3B,82,C
D,85,BD,11,45,82,4F68
240 DATA CD,85,BD,11,40,82,CD,88,BD,CD,2
3,82,C9,3E,03,21,56F9
250 DATA 59,82,BE,20,11,F1,FE,03,C0,21,3
6,82,11,3B,82,CD,5DE9
```

```
260 DATA 85,BD,CD,23,82,C9,3E,04,21,59,8
2,BE,20,32,F1,FE,65A3
270 DATA 03,C0,AF,21,03,00,11,45,82,CD,6
4,BD,21,40,82,CD,6BAF
280 DATA 9A,BD,21,36,82,11,36,82,CD,85,B
D,11,3B,82,CD,85,72D7
290 DATA BD,11,40,82,CD,85,BD,11,45,82,C
D,88,BD,CD,23,82,7AD2
300 DATA 3E,05,21,59,82,BE,20,63,F1,FE,0
4,C0,21,45,82,CD,81BA
310 DATA 9A,BD,AF,21,03,00,11,4A,82,CD,6
4,BD,11,36,82,21,8799
320 DATA 4F,82,CD,61,BD,11,3B,82,21,54,8
2,CD,61,BD,21,4F,8E75
330 DATA 82,11,4F,82,CD,85,BD,21,54,82,1
1,54,82,CD,85,BD,95D5
340 DATA 21,36,82,11,3B,82,CD,85,BD,11,4
F,82,CD,7C,BD,11,9C84
350 DATA 54,82,CD,7C,BD,11,40,82,CD,85,B
D,11,45,82,CD,85,A46C
360 DATA BD,11,4A,82,CD,88,BD,CD,23,82,C
9,3E,06,21,59,82,AB93
370 DATA BE,20,17,F1,FE,04,C0,21,36,82,1
1,3B,82,CD,85,BD,B2F1
380 DATA 11,40,82,CD,85,BD,CD,23,82,C9,3
E,07,21,59,82,BE,BA0D
390 DATA 20,21,F1,FE,03,C0,AF,21,03,00,1
1,40,82,CD,64,BD,C094
400 DATA 21,36,82,11,3B,82,CD,85,BD,11,4
0,82,CD,88,BD,CD,C7FC
410 DATA 23,82,C9,3E,08,21,59,82,BE,C0,F
1,FE,03,C0,21,40,CF3D
420 DATA 82,CD,9A,BD,21,36,82,11,36,82,C
D,85,BD,11,3B,82,D662
430 DATA CD,85,BD,11,40,82,CD,85,BD,CD,2
3,82,C9,01,05,00,DD94
440 DATA ED,B0,C9,21,36,82,ED,5B,30,82,0
1,05,00,ED,B0,C9,E539
450 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,E539
460 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,E539
470 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,E539
```

Chargeur ELECTRIC

```
10 d=&8000:f=&8225:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,0A,80,21,69,80,CD,D1,BC,C9,5
D,80,F5,FE,04,20,07AC
110 DATA 14,DD,66,01,DD,6E,00,22,0E,82,1
1,1A,82,CD,F7,81,0DF3
120 DATA DD,23,DD,23,3D,FE,03,20,14,DD,6
6,01,DD,6E,00,22,1416
130 DATA 0C,82,11,15,82,CD,F7,81,DD,23,D
D,23,3D,FE,02,20,1AEE
140 DATA 1A,DD,66,01,DD,6E,00,22,0A,82,1
1,10,82,CD,F7,81,212D
150 DATA DD,66,03,DD,6E,02,22,24,82,18,2
5,F1,C9,45,4C,45,2755
160 DATA 43,54,52,49,43,49,54,C5,00,00,0
0,00,00,00,00,00,2A2C
170 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,2A2C
180 DATA 3E,01,21,24,82,BE,20,18,F1,FE,0
3,C0,21,15,82,11,2FA3
190 DATA 15,82,CD,85,BD,11,10,82,EB,CD,8
5,BD,CD,FD,81,C9,38FA
200 DATA 3E,02,21,24,82,BE,20,17,F1,FE,0
4,C0,21,10,82,11,3E6D
210 DATA 15,82,CD,85,BD,11,1A,82,CD,88,B
D,CD,FD,81,C9,3E,4724
220 DATA 03,21,24,82,BE,20,17,F1,FE,04,C
0,21,10,82,11,15,4C6F
230 DATA 82,CD,7C,BD,11,1A,82,CD,85,BD,C
D,FD,81,C9,3E,04,5509
240 DATA 21,24,82,BE,20,31,F1,FE,04,C0,2
1,1F,82,CD,9A,BD,5C78
250 DATA 21,10,82,11,15,82,CD,88,BD,11,1
A,82,CD,85,BD,11,62B2
260 DATA 1F,82,CD,85,BD,3E,F9,CD,79,BD,2
1,10,82,11,10,82,69F2
270 DATA CD,7C,BD,CD,FD,81,C9,3E,05,21,2
4,82,BE,20,1F,F1,7204
```

```
280 DATA FE,04,C0,3E,01,CD,97,BD,21,10,8
2,CD,AF,BD,11,15,7938
290 DATA 82,CD,85,BD,11,1A,82,CD,85,BD,C
D,FD,81,C9,3E,06,81DD
300 DATA 21,24,82,BE,20,36,F1,FE,04,C0,2
1,10,82,11,10,82,87C1
310 DATA CD,85,BD,21,15,82,11,15,82,CD,8
5,BD,21,1A,82,11,8E0D
320 DATA 1A,82,CD,85,BD,21,15,82,11,1A,8
2,CD,85,BD,11,10,944D
330 DATA 82,EB,CD,7C,BD,CD,9D,BD,CD,FD,8
1,C9,3E,07,21,24,9D85
340 DATA 82,BE,20,43,F1,FE,04,C0,AF,21,0
1,00,11,1F,82,CD,A42B
350 DATA 64,BD,21,15,82,11,15,82,CD,85,B
D,21,1A,82,11,1A,A9A3
360 DATA 82,CD,85,BD,11,15,82,CD,85,BD,1
1,1F,82,EB,CD,88,B1DD
370 DATA BD,21,10,82,11,10,82,CD,85,BD,1
1,1F,82,CD,7C,BD,B8B7
380 DATA CD,9D,BD,CD,FD,81,C9,3E,08,21,2
4,82,BE,C0,F1,FE,C26C
390 DATA 03,C0,11,10,82,21,1A,82,CD,61,B
D,21,1A,82,11,15,C75D
400 DATA 82,CD,7C,BD,21,10,82,11,15,82,C
D,85,BD,11,1A,82,CDFC
410 DATA CD,88,BD,CD,FD,81,C9,01,05,00,E
D,B0,C9,21,10,82,D641
420 DATA ED,5B,0A,82,01,05,00,ED,B0,C9,0
0,00,00,00,00,00,DA81
430 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,DA81
440 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,DA81
```

Chargeur PHYSIQUE

```
10 d=&8000:f=&823A:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,0A,80,21,7E,80,CD,D1,BC,C9,7
5,80,F5,FE,05,20,07DA
110 DATA 14,DD,66,01,DD,6E,00,22,1E,82,1
1,2F,82,CD,05,82,0D55
120 DATA DD,23,DD,23,3D,FE,04,20,14,DD,6
6,01,DD,6E,00,22,1379
130 DATA 1C,82,11,2A,82,CD,05,82,DD,23,D
D,23,3D,FE,03,20,1986
140 DATA 14,DD,66,01,DD,6E,00,22,1A,82,1
1,25,82,CD,05,82,1EF3
150 DATA DD,23,DD,23,3D,FE,02,20,1A,DD,6
6,01,DD,6E,00,22,251B
160 DATA 18,82,11,20,82,CD,05,82,DD,66,0
3,DD,6E,02,22,39,2AAA
170 DATA 82,18,1D,F1,C9,50,48,59,53,49,5
1,55,C5,00,00,00,3013
180 DATA 00,00,21,24,82,BE,20,18,F1,FE,0
3,C0,21,15,82,11,354B
190 DATA AF,21,02,00,11,34,82,CD,64,BD,3
E,01,21,39,82,BE,3AAB
200 DATA 20,32,F1,FE,05,C0,21,2A,82,11,2
0,82,CD,85,BD,21,4161
210 DATA 20,82,11,20,82,CD,85,BD,11,25,8
2,CD,85,BD,11,34,47D1
220 DATA 82,CD,88,BD,11,2A,82,CD,7C,BD,1
1,2F,82,CD,7C,BD,4FF0
230 DATA CD,0B,82,C9,3E,02,21,39,82,BE,2
0,20,F1,FE,04,C0,56E0
240 DATA 3E,01,CD,97,BD,21,25,82,11,2A,8
2,CD,85,BD,CD,AC,5E4D
250 DATA BD,11,20,82,EB,CD,85,BD,CD,0B,8
2,C9,3E,03,21,39,6575
260 DATA 82,BE,20,1E,F1,FE,03,C0,21,25,8
2,11,25,82,CD,85,6C77
270 DATA BD,11,20,82,EB,CD,85,BD,11,34,8
2,CD,88,BD,CD,0B,7492
280 DATA 82,C9,3E,04,21,39,82,BE,20,30,F
1,FE,02,C0,21,25,7B00
```

```
290 DATA 82,CD,9A,BD,AF,21,0A,00,11,2A,8
2,CD,64,BD,21,20,816C
300 DATA 82,11,2A,82,CD,88,BD,CD,9D,BD,1
1,25,82,CD,85,BD,89AB
310 DATA 11,34,82,CD,85,BD,CD,0B,82,C9,3
E,05,21,39,82,BE,9081
320 DATA 20,19,F1,FE,03,C0,3E,01,CD,97,B
D,21,20,82,CD,AC,9808
330 DATA BD,11,25,82,CD,85,BD,CD,0B,82,C
9,3E,06,21,39,82,9ECF
340 DATA BE,20,1E,F1,FE,04,C0,21,25,82,1
1,20,82,CD,85,BD,A608
350 DATA 11,2A,82,CD,85,BD,11,20,82,EB,C
D,7C,BD,CD,0B,82,ADD2
360 DATA C9,3E,07,21,39,82,BE,20,20,F1,F
E,04,3E,01,CD,97,B450
370 DATA BD,C0,21,2A,82,CD,AF,BD,11,25,8
2,CD,85,BD,11,20,BBCB
380 DATA 82,EB,CD,85,BD,CD,0B,82,C9,3E,0
8,21,39,82,BE,C0,C40A
390 DATA F1,FE,04,C0,AF,21,9B,02,11,2F,8
2,CD,64,BD,21,20,CB1B
400 DATA 82,11,25,82,CD,85,BD,11,2A,82,C
D,88,BD,11,2A,82,D1F0
410 DATA CD,88,BD,11,2F,82,CD,85,BD,3E,F
3,21,20,82,CD,79,DA0D
420 DATA BD,CD,0B,82,C9,01,05,00,ED,B0,C
9,21,20,82,ED,5B,E164
430 DATA 18,82,01,05,00,ED,B0,C9,00,00,0
0,00,00,00,00,00,E46A
440 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,E46A
450 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,E46A
```

Chargeur MATHS

```
10 d=&8000:f=&823B:n=100:s=0
20 FOR i=1 TO 16
30 READ v$:v=VAL("&"+v$):s=s+v:POKE d,v:
d=d+1
40 NEXT
50 READ c$:c=VAL("&"+c$)
60 IF HEX$(c)<>HEX$(s) THEN PRINT"erreur
 ligne ";n:END
70 n=n+10:IF d<f THEN 20
80 PRINT"ok":END
100 DATA 01,0A,80,21,62,80,CD,D1,BC,C9,5
D,80,F5,FE,04,20,07A5
110 DATA 14,DD,66,01,DD,6E,00,22,10,82,1
1,1C,82,CD,F9,81,0DF2
120 DATA DD,23,DD,23,3D,FE,03,20,14,DD,6
6,01,DD,6E,00,22,1415
130 DATA 0E,82,11,17,82,CD,F9,81,DD,23,D
D,23,3D,FE,02,20,1AF3
140 DATA 1A,DD,66,01,DD,6E,00,22,0C,82,1
1,12,82,CD,F9,81,2138
150 DATA DD,66,03,DD,6E,02,22,35,82,18,2
5,F1,C9,4D,41,54,277D
160 DATA C8,00,00,00,00,00,05,82,DD,66,0
3,DD,6E,02,22,39,2BBA
170 DATA 82,18,1D,F1,C9,50,48,59,53,49,5
1,55,C5,00,00,00,3123
180 DATA 3E,01,21,35,82,BE,20,5F,F1,FE,0
4,C0,AF,21,04,00,36FE
190 DATA 11,30,82,CD,64,BD,11,12,82,21,2
1,82,CD,61,BD,11,3D14
200 DATA 17,82,21,26,82,CD,61,BD,11,1C,8
2,21,2B,82,CD,61,430C
210 DATA BD,21,26,82,11,17,82,CD,85,BD,2
1,21,82,11,2B,82,48CD
220 DATA CD,85,BD,11,30,82,EB,CD,85,BD,C
D,91,BD,11,26,82,516D
230 DATA CD,7C,BD,CD,94,BD,FE,FF,20,0F,1
1,30,82,21,12,82,5935
240 DATA CD,61,BD,CD,FF,81,C9,18,52,21,3
0,82,CD,9D,BD,11,61AB
250 DATA 17,82,21,26,82,CD,61,BD,21,12,8
2,11,12,82,CD,7C,679B
260 DATA BD,21,17,82,11,30,82,CD,7C,BD,1
1,12,82,CD,88,BD,6E92
270 DATA CD,91,BD,21,17,82,ED,5B,0E,82,0
1,05,00,ED,B0,21,7503
```

```
280 DATA 26,82,CD,91,BD,11,30,82,CD,7C,B
D,11,12,82,CD,88,7C89
290 DATA BD,ED,5B,10,82,01,05,00,ED,B0,C
9,3E,02,21,35,82,82A4
300 DATA BE,20,3E,F1,FE,02,C0,AF,21,01,0
0,11,37,82,CD,64,893D
310 DATA BD,11,12,82,21,17,82,CD,61,BD,2
1,17,82,11,37,82,8EC8
320 DATA CD,82,BD,DD,21,17,82,DD,CB,03,B
E,DD,7E,04,FE,81,97B2
330 DATA 28,0B,21,12,82,11,17,82,CD,85,B
D,18,DD,CD,FF,81,9E95
340 DATA C9,3E,03,21,35,82,BE,20,27,F1,F
E,04,C0,AF,21,02,A501
350 DATA 00,11,21,82,CD,64,BD,21,12,82,1
1,17,82,CD,7C,BD,AB08
360 DATA 11,1C,82,CD,85,BD,11,21,82,CD,8
8,BD,CD,FF,81,C9,B3A2
370 DATA 3E,04,21,35,82,BE,C0,F1,FE,04,C
0,AF,21,01,00,11,B9CF
380 DATA 21,82,CD,64,BD,11,17,82,21,26,8
2,CD,61,BD,21,17,BFF6
390 DATA 82,11,1C,82,CD,A0,BD,11,21,82,C
D,82,BD,21,26,82,C6DA
400 DATA 11,21,82,CD,82,BD,21,12,82,11,1
7,82,CD,85,BD,11,CD19
410 DATA 26,82,CD,88,BD,CD,FF,81,C9,01,0
5,00,ED,B0,C9,21,D576
420 DATA 12,82,ED,5B,0C,82,01,05,00,ED,B
0,C9,00,00,00,00,DA4C
430 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,DA4C
440 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,DA4C
450 DATA 00,00,00,00,00,00,00,00,00,00,0
0,00,00,00,00,00,DA4C
```

ANNEXE 3

VECTEURS SYSTÈME

VECTEURS MATHÉMATIQUES

Les vecteurs mathématiques figurent dans un tableau que vous trouverez au chapitre 2.

VECTEURS D'AFFICHAGE

Pour plus de précisions, voir chapitre 2, routine d'affichage de nombres entiers 16 bits en binaire.

— #C337 : routine d'affichage pour le CPC 464.
— #C380 : routine d'affichage pour le CPC 664.
— #C37D : routine d'affichage pour le CPC 6128.

VECTEURS CLAVIER

— #BB00 : Initialisation du clavier.

ENTREE : néant.
SORTIE : néant.

— #BB06 : Attente de la frappe d'une touche.

ENTREE : néant.
SORTIE : A code ASCII du caractère frappé.

VECTEURS GRAPHIQUES

Les adresses sont identiques quel que soit le CPC utilisé.

— #BB5A : Affichage d'un caractère.

ENTREE : A code ASCII du caractère à afficher.
SORTIE : néant.

— #BB90 : Fixe le numéro d'encre du stylo.

ENTREE : A de l'encre du stylo.
SORTIE : néant.

— #BBC9 : Fixe la position de l'origine.

ENTREE : De coordonnée absolue X.
HL coordonnée absolue Y.
SORTIE : néant.

— #BBCC : Indique la position de l'origine.

ENTREE : néant.
SORTIE : DE coordonnée absolue X.
HL coordonnée absolue Y.

— #BBCF : Placement des bords verticaux de la fenêtre graphique.

ENTREE : DE coordonnée premier bord.
HL coordonnée deuxième bord.
SORTIE : néant.

— #BBD2 : Placement des bords horizontaux de la fenêtre graphique.

ENTREE : DE coordonnée premier bord.
HL coordonnée deuxième bord.
SORTIE : néant.

— #BBDE : Fixe l'encre du stylo graphique.

ENTREE : A numéro de l'encre du stylo.
SORTIE : néant.

— #BBE4 : Fixe l'encre du papier graphique.

ENTREE : A numéro de l'encre du papier.
SORTIE : néant.

— #BBEA : Place un pixel à une position absolue.

ENTREE : DE coordonnée X du pixel.
HL coordonnée Y du pixel.
SORTIE : néant.

— #BBF0 : Indique le numéro d'encre d'un pixel.

ENTREE : DE coordonnée X du pixel.
HL coordonnée Y du pixel.
SORTIE : A numéro de l'encre du pixel.

— #BBF6 : Tracé d'une droite vers un point absolu.

ENTREE : DE coordonnée X du dernier point.
HL coordonnée Y du dernier point.
SORTIE : néant.

— #BBF9 : Tracé d'une droite vers un point relatif.

ENTREE : DE coordonnée X du dernier point.
HL coordonnée Y du dernier point.
SORTIE : néant.

— #BC11 : Indique le mode en cours.

ENTREE : néant.
SORTIE : A 0, 1 ou 2 suivant le cas.

— #BC14 : CLS de l'écran.

ENTREE : néant.
SORTIE : néant.

— #BC1D : Donne l'adresse mémoire d'un pixel.

ENTREE : DE coordonnée X du pixel.
HL coordonnée Y du pixel.
SORTIE : HL adresse mémoire.

— #BC29 : Donne l'adresse mémoire d'un octet situé dans la ligne précédente au-dessus de l'octet dont l'adresse est fournie.

ENTREE : HL adresse d'un octet.
SORTIE : HL adresse de l'octet au-dessus dans la ligne supérieure.

ANNEXE 4

LEXIQUE

Accumulateur :

Registre principal du processeur Z80, par lequel s'effectuent transferts et calculs.

Arobase : @

Caractère spécial qui, mis devant une variable, indique que l'on va traiter l'adresse de celle-ci.

Ascii :

Américain Standard Code for Information Interchange ; ce code permet de représenter les caractères sur 7 bits, ce qui permet de proposer une panoplie de 128 caractères en tout genre pour les micro-ordinateurs.

Assembleur :

Logiciel de traduction d'un programme écrit en Assembleur ; nom également donné au programme écrit dans ce langage.

Base :

Système de numération servant à exprimer les nombres ; en informatique sont utilisées les bases 2 (binaire), 10 (décimal), et 16 (hexadécimal).

Binaire :

Système de numération en base 2, qui utilise les symboles 0 et 1.

Bit :

Information élémentaire en informatique prenant les valeurs 0 ou 1 dans le système en base 2.

Buffer :

Zone de stockage d'informations utilisée dans un programme BASIC ou ASSEMBLEUR.

Cache :

Valeur permettant d'afficher les points d'un pixel, dans la couleur du STYLO, ou de celle du PAPIER.

Carry :

C'est l'un des bits du REGISTRE D'ETAT "F" ; sa fonction essentielle est d'indiquer un dépassement éventuel, suite à une OPERATION ARITHMETIQUE ou LOGIQUE.

Chaîne :

C'est une suite de caractères ASCII, repérée en BASIC par le signe $.

Chargeur :

Il s'agit d'un programme permettant de mémoriser une suite de codes à un emplacement défini.

Code objet :

Programme en binaire directement exécutable par la machine.

Code source :

Programme compréhensible par l'homme et le logiciel ASSEMBLEUR. Ce programme n'est pas directement exécutable par la machine.

Décalage :

C'est la translation d'un bit vers la droite ou vers la gauche, dans un registre donné. Dans ce cas, le premier ou le dernier bit va dans le REPORT.

Décrémentation :

Soustraction de 1 d'un registre, d'un octet donné, ou d'une variable BASIC.

Dépassement :

C'est la conséquence d'une opération dont la valeur du résultat dépasse la valeur maximale autorisée ; soit 255 pour un traitement sur 8 bits, et 65536 pour un traitement sur 16 bits.

Déplacement relatif :

Se dit d'un saut, à l'intérieur d'un programme assembleur, s'effectuant à partir de l'instruction elle-même ; celui-ci peut aller de −127 à +128 octets.

Désassembleur :

Logiciel effectuant la traduction d'un programme binaire en langage ASSEMBLEUR.

Digit :

Equivalent du bit dans le système en base 2.

Données :

Informations en entrée ou en sortie d'un programme en ASSEMBLEUR.

Dump :

Logiciel de visualisation d'une portion de mémoire en codes hexa et en codes ASCII.

Entier :

Valeur numérique sur 8 ou 16 bits en ASSEMBLEUR.

Etiquette :

C'est ce qui permet de repérer une instruction assembleur, en vue d'un branchement par une instruction de saut.

Exposant :

Octet d'un nombre flottant permettant de faire la séparation entre partie entière et partie décimale.

Fenêtre :

Cadre rectangulaire défini dans la zone d'affichage de l'écran.

Flag :

Equivalent d'INDICATEUR dans le REGISTRE D'ETAT, reflétant le résultat d'une opération quelconque.

Flottant :

C'est la représentation en mémoire d'un nombre réel, configuré sur 5 octets, dont 4 pour la mantisse, et 1 pour l'exposant.

Fonction booléenne :

Opération logique s'effectuant entre 2 octets ou 1 octet et une valeur donnée.

Hexadécimal :

Système de numération en base 16, utilisant les symboles 0 à 9 et A à F.

Incrémentation :

Addition de 1 d'un registre, d'un octet donné, ou d'une variable BASIC.

Indicateur :

Equivalent de FLAG dans le REGISTRE D'ETAT.

Indice :

Repère du niveau hiérarchique d'un tableau BASIC défini en DIM.

Interpréteur BASIC :

Logiciel procédant à la traduction du BASIC en langage exécutable par la machine.

Interruption :

Elément interne ou externe à un programme, provoquant l'arrêt momentané de celui-ci, en vue d'effectuer un autre traitement.

Label :

Equivalent d'ETIQUETTE dans un programme ASSEMBLEUR.

Mantisse :

Partie d'un flottant représentant les chiffres significatifs.

Octet :

Groupe de 8 bits permettant de représenter des nombres entre 0 et 255.

Opérations arithmétiques :

Ce sont les opérations classiques : +, −, ×, /

Opérations logiques :

Ce sont les fonctions de type AND, OR, XOR.

Overflow :

Equivalent de DEPASSEMENT, suite à un traitement arithmétique ou logique.

Papier :

Feuille virtuelle sur laquelle s'inscrivent graphisme et texte, dont le nombre de couleurs possibles dépend du mode.

Pixel :

Unité de base de l'écran, dont le nombre de points qui la forment dépend du mode écran en cours.

Poids :

C'est la valeur correspondante à 2 puissance « numéro du bit considéré », en sachant que dans un octet le numérotage des bits va de 0 à 7, le premier bit étant le bit 0.

Pointeur :

Valeur permettant de pointer sur une zone quelconque de la mémoire, en vue d'y accéder facilement.

Pointeur de pile :

Se nomme aussi STACK POINTEUR ; permet au système de gérer une zone dans laquelle sont stockées des données sur 2 octets ; ceci est utilisé dans le cas de la sauvegarde des registres.

Quartet :

C'est la partie droite ou gauche d'un octet, soit sur 4 bits.

Ram :

De l'anglais Random-Access Memory, c'est-à-dire mémoire utilisateur.

Registre :

Un registre est constitué d'un octet pour les registres simples, et de 2 octets pour les registres doubles ; nous pouvons dénombrer les registres suivants :

— registres simples : A, B, C, D, E, H, L, I R
— registres doubles : AF, BC, DE, HL
— registre d'état : F
— registres d'index : IX,IY
— registres auxiliaires : A', F', B', C', D', E', H', L' ; ceux-ci sont utilisés en double registre et permettent la sauvegarde des registres doubles courants.

Report :

Equivalent de CARRY, dans le REGISTRE D'ETAT.

Rom :

De l'anglais Read-Only Memory, c'est-à-dire une mémoire qui ne peut être que lue.

Rotation :

Translation des bits d'un octet à gauche ou à droite, avec récupération du bit 0 dans le bit 7, ou du bit 7 dans le bit 0.

RSX :

De l'anglais Résident System eXtension, c'est-à-dire extension d'instructions BASIC implantée en RAM.

Scrolling :

Déplacement d'une zone de l'écran, ou de sa totalité, dans une direction donnée.

Système d'exploitation :

Logiciel gérant le déroulement des différentes tâches d'un ordinateur.

Vecteur :

Adresse (en RAM) de branchement à une routine située en ROM.

Zoom :

Agrandissement d'une zone de l'écran.

ANNEXE 5

LA DISQUETTE PROGRAMME

Vous pouvez commander la disquette d'accompagnement de cet ouvrage. Le sommaire en est le suivant :

Face 1 : — Programme expliquant l'utilisation et décrivant le contenu de la disquette.

— Tous les programmes listés dans l'ouvrage sous la forme de fichiers binaires. Ceux-ci sont récupérables soit directement en faisant suivre l'ordre LOAD de l'adresse de chargement, soit à l'aide d'un logiciel d'assemblage-désassemblage permettant de modifier la routine.

— Programme de démonstration des routines se présentant sous la forme de menus permettant de sélectionner le programme désiré.

— Page écran appelée « ECRAN.SCR » formée d'un dessin utilisé pour les démonstrations des routines graphiques.

Face 2 : — Les CHARGEURS de routines permettant leur installation en mémoire par l'intermédiaire d'une série de DATA. Cette méthode permet de modifier les routines sans l'intervention d'un assembleur-désassembleur.

NOTE IMPORTANTE

Les programmes fournis avec la disquette d'accompagnement sont ceux qui sont présentés dans l'ouvrage ; soit des programmes qui fonctionnent, dans la majorité, sur le CPC 6128.
Pour l'adaptation sur CPC 464, se reporter aux notes indiquées dans l'ouvrage, pour chacune des routines traitées.
En ce qui concerne l'adaptation pour le CPC 664, se reporter à l'annexe III, pour les équivalences de vecteurs.

Vente par correspondance

BON DE COMMANDE

Je commande la disquette d'accompagnement du livre : « Assembleur et routines RSX » au prix de 150,00 FF*.

Renvoyez-nous ce bon rempli (découpé ou copié), avec votre règlement, plus de 10,00 FF** de frais de port et d'emballage, par chèque bancaire ou postal établi à l'ordre de LA CONSOLE.
Adresse : La Console, 5, place du Colonel-Fabien, 75010 Paris

Nom ..... Prénom .....
Adresse ....
..... Ville .....
Code postal ..... Pays .....

* Prix valable jusqu'au 31 décembre 1986.
** Hors CEE, prévoyez 15,00 FF pour supplément de frais d'expédition.

CONSEILS DE LECTURE

Pour approfondir vos connaissances en BASIC, mieux connaître le système des CPC 464, 664 et 6128, et maîtriser le graphisme sur Amstrad, P.S.I. vous propose une palette d'ouvrages utiles.

POUR MAÎTRISER LE BASIC AMSTRAD

— **Basic Amstrad 1 — Méthodes pratiques** — Jacques Boisgontier et Bruno Césard (Editions du P.S.I.)

Pour ceux qui ont déjà pratiqué un Basic, voici un ouvrage de perfectionnement au Basic Amstrad. Un chapitre sur le CP/M 2.2. et le CP/M Plus donne les principales commandes systèmes.

— **Basic Amstrad 2 — Programmes et fichiers** — Jacques Boisgontier (Editions du P.S.I.)

Pour pratiquer le Basic Amstrad, cet ouvrage donne de nombreux programmes de gestion, d'éducation et de jeu où le rôle des fichiers est expliqué et largement commenté.

— **Basic plus — 80 routines sur Amstrad** — Michel Martin (Editions du P.S.I.)

Pour pousser votre Amstrad au maximum de ses capacités : 80 routines de simulation d'instructions qui n'existent pas en Basic Amstrad.

POUR MIEUX CONNAÎTRE LE SYSTÈME DES CPC

— **Clefs pour Amstrad 1 — système de base** — Daniel Martin (Editions du P.S.I.)

Mémento présentant synthétiquement le jeu d'instructions du Z80, les points d'entrée des routines système, les connecteurs et brochages, etc.

Le livre de chevet du programmeur sur Amstrad.

— **Clefs pour Amstrad 2 — système disque** — Daniel Martin et Philippe Jadoul (Editions du P.S.I.)

Ce deuxième tome consacré au système disque présente les points d'entrée des routines disque, les blocs de contrôle, la programmation et les brochages des circuits spécialisés...

La deuxième partie du livre est aussi destinée aux possesseurs d'Amstrad 8256.

— **CP/M plus sur Amstrad** — Yvon Dargery (Editions du P.S.I.)

Toutes les commandes CP/M et CP/M plus pour maîtriser le système des 6128 et 8256 : un ouvrage de référence illustré par de nombreux programmes.

— **Le livre de l'Amstrad — Tome 1** — Daniel Martin et Philippe Jadoul (BCM — diffusé par P.S.I.)

Ce livre, destiné aux programmeurs des CPC 464, 664 et 6128, donne une étude complète de tous les circuits internes, et analyse la structure interne du BASIC. Vous y trouverez, en outre, une étude complète des RSX, et des programmes de scrolling, de traçage de rectangles, de coloriage de surface et de manipulation vectorielle.

POUR CONCEVOIR ET AMÉLIORER VOS GRAPHISMES

— **Mathématiques et graphismes** — Gérald Grandpierre et Richard Cotté (Editions du P.S.I.)

De très beaux graphismes sont générés par des équations mathématiques. L'univers des fractals, les déformations et les enveloppes, les surfaces en Z2 sont étudiées dans ce livre très pédagogique et de haut niveau. Tous les programmes, écrits en Basic standard, sont facilement adaptables au Basic Amstrad.

— **Création et animations graphiques sur Amstrad CPC** — Gilles Fouchard et Jean-Yves Corre (Editions du P.S.I.)

Dessiner avec la souris ou le joystick et apprendre à faire des scrolling, à fabriquer une gomme, à inverser une image ou à l'éclater en une myriade de points, tel est l'objectif de ce livre écrit en assembleur Amstrad.

— **Graphisme en assembleur sur Amstrad CPC** — Francis Piérot (Editions du P.S.I.)

Cet ouvrage met en pratique les notions d'assembleur acquises par le lecteur, grâce à de nombreux programmes commentés. Du simple tracé d'un cercle, à l'animation d'une soucoupe volante dans un décor, en passant par la mise en mouvement d'une corne d'abondance, vous apprendrez à maîtriser l'assembleur et à créer de très belles pages-écran.

Achevé d'imprimer par l'imprimerie Tardy Quercy S.A. - Bourges
D.L. : Août 1986 - N° d'Imprimeur : 13152 - N° d'Édition : 86595-352-1

Votre avis nous intéresse

Pour nous permettre de faire de meilleurs livres, adressez-nous vos critiques sur le présent ouvrage.

— ***Ce livre vous donne-t-il toute satisfaction ?***

.....
.....
.....
.....
.....

— ***Y a-t-il un aspect du problème que vous auriez aimé voir abordé ?***

.....
.....
.....
.....
.....

Si vous souhaitez des éclaircissements techniques, écrivez-nous, nous ne manquerons pas de vous répondre directement.

Où avez-vous acheté ce livre ?

☐ cadeau ☐ librairie ☐ autres
☐ exposition ☐ boutique micro

Comment en avez-vous eu connaissance ?

☐ publicité ☐ catalogue ☐ autres
☐ exposition ☐ conseils d'un ami

Avez-vous déjà acquis des livres P.S.I. ?

Lesquels ? .....
.....
qu'en pensez-vous ? .....
.....
.....

Nom ..... Prénom ..... Age.....
Adresse .....
Profession .....
Centre d'intérêt .....

CATALOGUE GRATUIT

Vous pouvez obtenir un catalogue complet des ouvrages PSI, sur simple demande, ou en retournant cette page remplie à votre libraire, à votre boutique micro ou aux

Editions du PSI
BP 86
77402 Lagny-sur-Marne Cedex

LANGAGE ET DÉVELOPPEMENT

RSX ET ROUTINES ASSEMBLEUR SUR AMSTRAD CPC

Cet ouvrage s'adresse à vous, possesseurs d'Amstrad CPC 464, 664 et 6128, qui souhaitez mettre en pratique vos connaissances en assembleur Z80.

Après avoir découvert le jeu d'instructions du Z80, vous entrerez dans le vif du sujet grâce à de courts programmes utiles qui vous seront commentés, expliqués et fournis avec leur organigramme.

Vous maîtriserez rapidement le traitement des listes de caractères, l'affichage et la conversion des nombres. Vous découvrirez également les possibilités graphiques de votre machine, et vous effectuerez des calculs en créant des fichiers de formules.

En majorité, les programmes proposés peuvent compléter utilement vos programmes Basic, sous forme de nouvelles instructions de type RSX et vous ouvriront ainsi de larges possibilités de créations personnelles.

9 782865 953523

ÉDITIONS DU P.S.I.
BP 86 - 77402 LAGNY S/MARNE CEDEX - FRANCE

ISBN 2-86595-352-1 200 FF

PHOTO L. MAUNOURY

Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE

ACME

https://acpc.me/